

ActuateOne™

One Design
One Server
One User Experience

Information Console Developer Guide

Information in this document is subject to change without notice. Examples provided are fictitious. No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of Actuate Corporation.

© 1995 - 2013 by Actuate Corporation. All rights reserved. Printed in the United States of America.

Contains information proprietary to:

Actuate Corporation, 951 Mariners Island Boulevard, San Mateo, CA 94404

www.actuate.com

The software described in this manual is provided by Actuate Corporation under an Actuate License agreement. The software may be used only in accordance with the terms of the agreement. Actuate software products are protected by U.S. and International patents and patents pending. For a current list of patents, please see <http://www.actuate.com/patents>.

Actuate Corporation trademarks and registered trademarks include:

Actuate, ActuateOne, the Actuate logo, Archived Data Analytics, BIRT, BIRT 360, BIRT Analytics, The BIRT Company, BIRT Data Analyzer, BIRT iHub, BIRT Performance Analytics, Collaborative Reporting Architecture, e.Analysis, e.Report, e.Reporting, e.Spreadsheet, Encyclopedia, Interactive Viewing, OnPerformance, The people behind BIRT, Performancesoft, Performancesoft Track, Performancesoft Views, Report Encyclopedia, Reportlet, X2BIRT, and XML reports.

Actuate products may contain third-party products or technologies. Third-party trademarks or registered trademarks of their respective owners, companies, or organizations include:

Mark Adler and Jean-loup Gailly (www.zlib.net): zlib. Adobe Systems Incorporated: Flash Player. Amazon Web Services, Incorporated: Amazon Web Services SDK, licensed under the Apache Public License (APL). Apache Software Foundation (www.apache.org): Ant, Axis, Axis2, Batik, Batik SVG library, Commons Command Line Interface (CLI), Commons Codec, Crimson, Derby, Hive driver for Hadoop, Pluto, Portals, Shindig, Struts, Tomcat, Xalan, Xerces, Xerces2 Java Parser, and Xerces-C++ XML Parser. Castor (www.castor.org), ExoLab Project (www.exolab.org), and Intalio, Inc. (www.intalio.org): Castor. Day Management AG: Content Repository for Java. Eclipse Foundation, Inc. (www.eclipse.org): Babel, Data Tools Platform (DTP) ODA, Eclipse SDK, Graphics Editor Framework (GEF), Eclipse Modeling Framework (EMF), and Eclipse Web Tools Platform (WTP), licensed under the Eclipse Public License (EPL). Gargoyle Software Inc.: HtmlUnit, licensed under Apache License Version 2.0. GNU Project: GNU Regular Expression, licensed under the GNU Lesser General Public License (LGPLv3). HighSlide: HighCharts. Jason Hsueh and Kenton Varda (code.google.com): Protocol Buffer. IDAutomation.com, Inc.: IDAutomation. IDRsolutions Ltd.: JBIG2, licensed under the BSD license. InfoSoft Global (P) Ltd.: FusionCharts, FusionMaps, FusionWidgets, PowerCharts. Matt Inger (sourceforge.net): Ant-Contrib, licensed under Apache License Version 2.0. Matt Ingenthron, Eric D. Lambert, and Dustin Sallings (code.google.com): Spymemcached, licensed under the MIT OSI License. International Components for Unicode (ICU): ICU library. jQuery: jQuery, licensed under the MIT License. Yuri Kanivets (code.google.com): Android Wheel gadget, licensed under the Apache Public License (APL). LEAD Technologies, Inc.: LEADTOOLS. The Legion of the Bouncy Castle: Bouncy Castle Crypto APIs. Bruno Lowagie and Paulo Soares: iText, licensed under the Mozilla Public License (MPL). Microsoft Corporation (Microsoft Developer Network): CompoundDocument Library. Mozilla: Mozilla XML Parser, licensed under the Mozilla Public License (MPL). MySQL Americas, Inc.: MySQL Connector. Netscape Communications Corporation, Inc.: Rhino, licensed under the Netscape Public License (NPL). OOPS Consultancy: XMLTask, licensed under the Apache License, Version 2.0. Oracle Corporation: Berkeley DB, Java Advanced Imaging, JAXB, JDK, Jstl. PostgreSQL Global Development Group: pgAdmin, PostgreSQL, PostgreSQL JDBC driver. Progress Software Corporation: DataDirect Connect XE for JDBC Salesforce, DataDirect JDBC, DataDirect ODBC. Rogue Wave Software, Inc.: Rogue Wave Library SourcePro Core, tools.h++. Sam Stephenson (prototype.conio.net): prototype.js, licensed under the MIT license. Sencha Inc.: Ext JS, Sencha Touch. ThimbleWare, Inc.: JMemcached, licensed under the Apache Public License (APL). World Wide Web Consortium (W3C) (MIT, ERCIM, Keio): Flute, JTIty, Simple API for CSS. XFree86 Project, Inc.: (www.xfree86.org): xvfb. ZXing authors (code.google.com): ZXing, licensed under the Apache Public License (APL).

All other brand or product names are trademarks or registered trademarks of their respective owners, companies, or organizations.

Document No. 130131-2-640301 January 23, 2013

Contents

About <i>Information Console Developer Guide</i>	vii
---	------------

Part 1

Customizing Actuate Information Console

Chapter 1

Introducing Actuate Information Console	3
--	----------

About Actuate Information Console	4
Setting up Actuate Information Console	5
Generating a web archive (WAR) for installation	5
Understanding Actuate Information Console load balancing	6
Deploying a load balancer for an Actuate BIRT iHub cluster	7
About using a cluster of application servers	7
About Actuate Information Console architecture	8
Using proxy servers with Actuate Information Console	8
About Actuate Information Console pages	10
Working with Actuate Information Console URIs	11
About Actuate Information Console URIs	11
Using a special character in a URI	12
About UTF-8 encoding	14
About Actuate Information Console functionality levels	14
Customizing functionality levels	16
Customizing functionality level features	18
Preserving functionality levels and features	20

Chapter 2

Creating a custom Information Console web application	21
--	-----------

Information Console web application structure and contents	22
Understanding Information Console directory structure	23
Building a custom Information Console context root	27
Activating a new or custom web application	29
Configuring a custom Information Console web application	29
Customizing Information Console configuration	30
Setting the default locale	30
Controlling the Message Distribution service load balancing	31
Specifying the default Encyclopedia volume and server	32
Modifying text and messages	33
Customizing Information Console text and messages	34
Customizing Actuate BIRT iHub error messages	36

Customizing an Information Console web application	39
Modifying the landing page	40
Viewing modifications to a custom web application	41
Locating existing pages and linking in new pages	42
Obtaining information about the user and the session	43
Customizing accessible files and page structure using templates	44
Specifying a template and template elements	45
About the dashboard template	46
Changing a template	46
Modifying existing content or creating new content	48
Modifying global style elements	49
Customizing Actuate Information Console using skins	49
Using skins	49
Managing skins using the skin manager	51
Customizing and cloning skins	52
Understanding style definition files	56
Specifying colors and fonts	57
Customizing page styles for BIRT Studio	58
Modifying graphic images	59

Part 2

Actuate Information Console reference

Chapter 3

Actuate Information Console configuration	63
About Information Console configuration	64
Configuring the Information Console web application	64
Configuring Information Console using web.xml	64
Configuring Information Console using volumeProfile.xml	70
Using a volume profile defined in volumeProfile.xml	71
Overriding the volume specified in a volume profile	71
Understanding temporary volume profiles	72
Configuring Information Console functionality levels with functionality-level.config	72
Configuring Information Console locales	75
Configuring Information Console time zones	76
Customizing messages and text according to locale	76
Configuring Shindig 2.0 for a WAR or EAR deployment	78
Configuring the connection to iHub	79
Configuring the BIRT Viewer and Interactive Viewer	80
Configuring BIRT Studio	80
Configuring BIRT Data Analyzer	80

Chapter 4

Actuate Information Console URIs 81

Actuate Information Console URIs overview	82
Actuate Information Console URIs quick reference	82
Common URI parameters	84
Information Console Struts actions	85
Actuate Information Console URIs reference	92
about page	95
banner page	95
browse file page	96
calendar page	96
channels page	96
completed request page	97
create folder page	98
dashboard page	98
delete file status page	98
delete job page	99
delete status page	99
detail page	100
drop page	102
error page	103
execute report page	103
general options page	106
home page	107
index page	108
license page	110
list page	111
login banner page	114
login page	114
logout page	115
My dashboard page	115
notification page	116
options page	117
output page	118
page not found page	119
parameters page	120
pending page	120
ping page	120
privileges page	123
running page	123
schedule page	124
scheduled job page	125
search folders page	125

submit job page	126
Actuate BIRT Viewer URIs reference	130

Chapter 5

Actuate Information Console JavaScript 131

Actuate Information Console JavaScript overview	132
Actuate Information Console JavaScript reference	132

Chapter 6

Actuate Information Console servlets 135

Information Console Java servlets overview	136
About the base servlet	136
Invoking a servlet	136
Information Console Java servlets reference	137
DownloadFile servlet	137
Interactive Viewer servlet	137

Chapter 7

Actuate Information Console custom tags 141

Information Console custom tag overview	142
Information Console custom tags quick reference	142
Information Console custom tag libraries	142
Information Console custom tags	143
Information Console custom tags reference	144
bundle	144
content	145
copyFileFolder	146
formatDate	146
login	147
message	149
tab	150
tabBegin	151
tabEnd	151
tabMiddle	152
tabMiddleSelected	153
tabPanel	153
tabSeparator	155

Chapter 8

Actuate Information Console JavaBeans 157

Information Console JavaBeans overview	158
Information Console JavaBeans package reference	158
Information Console JavaBeans class reference	158

Channels	158
Documents	159
General	160
Jobs	160
Skins	161
Users	162
Information Console UserInfoBean class reference	162

Chapter 9

Using Actuate Information Console security 169

About Actuate Information Console security	170
Protecting corporate data	170
Protecting corporate data using firewalls	170
Protecting corporate data using Network Address Translation	171
Protecting corporate data using proxy servers	171
Understanding the authentication process	171
Creating a custom security adapter	172
Accessing the IPSE Java classes	173
Creating a custom security adapter class	173
Deploying a custom security adapter	174
Understanding the security adapter class	175
Creating an upload security adapter	178
Accessing the necessary Java classes	179
Creating a custom security adapter class	179
Deploying an upload security adapter	180
Understanding the upload security adapter interface	181

Chapter 10

Customizing Information Console online help 183

About Actuate Information Console online help files	184
Understanding the help directory structure	184
Understanding a help collection	185
Understanding a document root	186
Understanding context-sensitive help	187
Understanding locale support	188
Using a custom help location	189
Creating a localized help collection	191
Customizing icons, links, and the company logo	193
Changing the corporate logo	193
Changing the additional links footer in help content pages	194
Changing the Google translate element in help content pages	196
Changing icons	196
Changing the browser window title	198

Changing help content198

 Changing existing help content198

 Adding or removing help topics199

 Adding and removing content files200

 Changing the table of contents201

 Changing the index204

Index 207

About Information Console Developer Guide

Information Console Developer Guide is a guide to designing, deploying, and accessing custom reporting web applications using Actuate Information Console.

Information Console Developer Guide includes the following chapters:

- *About Information Console Developer Guide.* This chapter provides an overview of this guide.
- *Part 1. Customizing Actuate Information Console.* This part describes how to use Information Console and how to customize its appearance and layout.
- *Chapter 1. Introducing Actuate Information Console.* This chapter introduces Actuate Information Console web applications and explains how Information Console works.
- *Chapter 2. Creating a custom Information Console web application.* This chapter explains how to work with Information Console JSP files to design custom reporting web applications.
- *Part 2. Actuate Information Console reference.* This part describes the code components that make up Information Console, such as URIs, JavaScript files, servlets, tags, beans, and security facilities.
- *Chapter 3. Actuate Information Console configuration.* This chapter describes the Information Console configuration files and parameters.
- *Chapter 4. Actuate Information Console URIs.* This chapter describes the Information Console JSPs and URL parameters.
- *Chapter 5. Actuate Information Console JavaScript.* This chapter describes the Information Console JavaScript files.
- *Chapter 6. Actuate Information Console servlets.* This chapter describes the Information Console Java servlets.

- *Chapter 7. Actuate Information Console custom tags.* This chapter describes the Information Console custom tag libraries.
- *Chapter 8. Actuate Information Console JavaBeans.* This chapter lists the Information Console JavaBeans.
- *Chapter 9. Using Actuate Information Console security.* This chapter introduces the Information Console Security Extension (IPSE) and explains how to use it.
- *Chapter 10. Customizing Information Console online help.* This chapter describes how to customize the Information Console online help files.

Part One

Customizing Actuate Information Console

1

Introducing Actuate Information Console

This chapter contains the following topics:

- About Actuate Information Console
- About Actuate Information Console architecture

About Actuate Information Console

Actuate Information Console is a web application that supports accessing and working with report information using a web browser. Web developers and designers use Actuate Information Console's industry-standard technology to design custom e.reporting web applications that meet business information delivery requirements.

Actuate Information Console technology is platform independent and customizable. By separating user interface design from content generation, Information Console ensures that reporting web application development tasks can proceed simultaneously and independently. You deploy Actuate Information Console on a network with Actuate BIRT iHub. Information Console accesses and stores documents on an Encyclopedia volume managed by iHub. Actuate Information Console technology is also scalable and supports clustering. On a Windows system, the default context root for Information Console is C:\Program Files\Actuate\iPortal2\iportal for Information Console installed separately or C:\Program Files\Actuate\iHub2\servletcontainer\iportal for Information Console embedded in the BIRT iHub application. On a UNIX-based system, the default context root for Information Console is \$Home/iPortal2/iportal for Information Console installed separately or \$Home/iHub2/servletcontainer/iportal for Information Console embedded in the BIRT iHub application.

Actuate Information Console technology includes the following features:

- JavaServer Pages (JSPs) support creating HTML or XML pages that combine static web page templates with dynamic content.
- Distributing requests to multiple Actuate BIRT iHub machines in an Actuate BIRT iHub System cluster balances server loads.
- Simple Object Access Protocol (SOAP) standards provide plain text transmission of XML using HTTP.
- Actuate Information Delivery API supports direct communication between the pages' custom tags and Actuate BIRT iHub.
- The full range of authentication and authorization functionality that Actuate BIRT iHub provides is available.
- Secure HTTP (HTTPS) supports secure information transfer on the web.
- Licensed options on BIRT iHub provide additional functionality. To use these options on a BIRT iHub System, the BIRT iHub System must be licensed for the options. For example, to use browser-based tools, such as BIRT Interactive Viewer or BIRT Data Analyzer, the BIRT iHub requires the appropriate license options.

The BIRT 360 option for BIRT iHub is required to use dashboard and gadget files. If these options are not available, users cannot open dashboards or gadgets in Information Console.

Setting up Actuate Information Console

You install Information Console in either of two ways:

- As a separate web application. This method enables native load balancing for iHub clusters, redundancy to support constant report services over the web, and secure networks using firewalls and proxy servers as described in Chapter 9, “Using Actuate Information Console security.”
- Automatically on the same host with iHub. This method provides reports locally on each iHub machine.

For enterprise architectures, installing Information Console on several web servers is recommended.

To deploy a report to the web, you need:

- An Actuate Information Console installation.
- An application server or JSP or servlet engine such as Actuate embedded servlet engine or IBM WebSphere.
- One or more Actuate designer tools and Actuate BIRT iHub System with Actuate Management Console.
- Actuate BIRT iHub administrator privileges.
- Permission to read, write, and modify operating system directories as necessary. For example, the directory Java uses to hold temporary files is defined by the `java.io.tmpdir` property and is by default the value of the TMP system variable in the Windows environment and `/var/tmp` in the UNIX and Linux environments. Read and write permission must be provided to the application server running Information Console for this directory.

This section discusses deployment concerns that may affect your Information Console installation and how you wish to deploy reports to the web. For more information about installing Information Console, see *Installing BIRT iHub for Windows* or *Installing BIRT iHub for Linux*.

Generating a web archive (WAR) for installation

To deploy Information Console on an application server, you can use a WAR file of your Information Console application. Generating Web Archive is a feature of Actuate Information Console that is available to Administrator-level users. This feature creates a WAR file of your entire Actuate Information Console system. Information Console streams the WAR file to your browser. You select a file name and location to save the file. After you customize your system, you can create a

WAR file to deploy the customized Information Console on other machines. The customizations can include any modifications of JavaScript, JavaServer Pages (JSPs) and other web pages, and skins. Later chapters in this book provide detailed information about customizing JavaScript and JSPs.

If Actuate Information Console is deployed as a WAR file, you cannot further customize skins, add pages, or make any other changes that affect the Actuate Information Console file structure in the WAR file. Instead, install Actuate Information Console as a directory structure with the installation wizard on your product CD and make your changes to that installation. Then use Generate Web Archive to create a new WAR file and deploy that WAR file to your application server.

How to customize and deploy Actuate Information Console in a cluster

To customize Actuate Information Console and deploy it to application servers in a clustered environment, use the following general procedure.

- 1 Install Actuate Information Console on one of the machines in your cluster.
- 2 Customize the Actuate Information Console JavaScript, skins, and web pages as desired.
- 3 Open Information Console. On the landing page, choose My Documents.
- 4 Log in as an administrator-level user. On the Information Console banner, choose Customization.
- 5 Choose Generate Web Archive. At the prompt, provide a location for the WAR file. For example, provide the location where your application server accesses WAR files. By default, the name of the WAR file of your customized Actuate Information Console installation is `acweb.war`.
- 6 Deploy the WAR file to each remaining machine in your cluster.

Understanding Actuate Information Console load balancing

Actuate Information Console supports two kinds of load balancing, as illustrated in Figure 1-1, to ensure high availability and to distribute tasks for efficient processing:

- Actuate Message Distribution service (MDS) balances the request load among Actuate BIRT iHub machines in an Actuate BIRT iHub cluster.
The Message Distribution service eliminates the need for a third-party network load balancer in front of the Actuate BIRT iHub tier. Actuate Information Console determines which machines in a cluster have MDS running and detects when the MDS machines go offline. MDS distributes the load among the available servers and does not attempt to send a request to an offline machine.

- Clustered Actuate Information Console machines can use a third-party application to balance the load among the application servers.

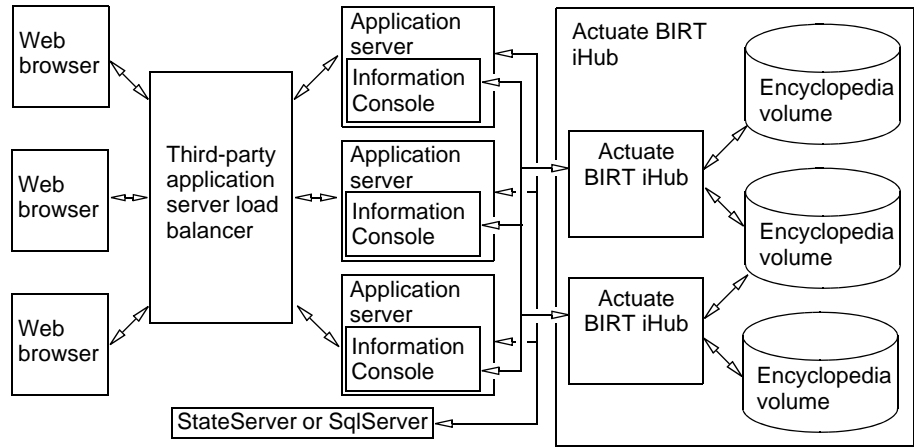


Figure 1-1 Load-balancing architecture for Information Console

Deploying a load balancer for an Actuate BIRT iHub cluster

To deploy a load balancer or proxy layer in front of the Actuate BIRT iHub tier, disable the Actuate load-balancing support by setting the `MDS_ENABLED` configuration parameter to `False` in the `web.xml` Actuate Information Console configuration file.

About using a cluster of application servers

If the application servers running Information Console support session state management, you can configure Actuate Information Console and the application servers to share and maintain a web browsing session state across a cluster of Information Console instances. Configuring the application servers to track the state of each Information Console instance supports reusing authentication information. In other words, you can log in to an Information Console instance and send a request using another Information Console instance without logging in again using the second instance.

If you do not use an application server to track session state information, managing the session state is fast, but you lose a user's state information when you restart Actuate Information Console or your application server.

Sharing session state information takes advantage of the application servers' failover features. If a user is on a cluster application server running Information Console and that application server fails, another application server running Information Console can manage the user's session.

An application server works with one or more database servers to manage session state information. All application servers must have access to the database server to store and retrieve session state information. For specific information about configuring your installation, see your application server documentation.

About Actuate Information Console architecture

This section describes the general operation, authentication, and structure of Information Console as a web application.

The Actuate Information Console architecture is illustrated in Figure 1-2.

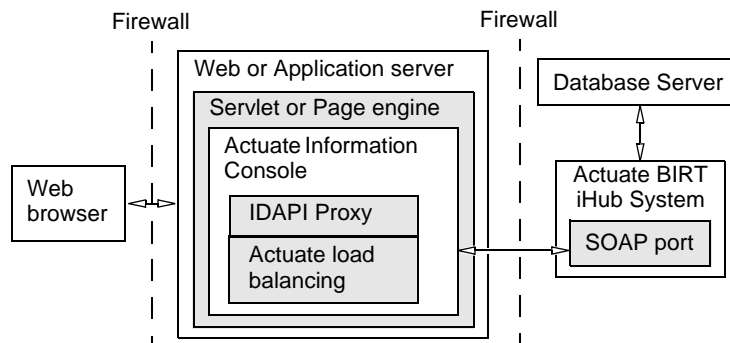


Figure 1-2 Actuate Information Console architecture overview

A user submits a request by choosing a link on a web page that specifies an Actuate Information Console URI. As shown in Figure 1-2, the web or application server receives the URI as an HTTP request and passes the request to the servlet or page engine. The engine invokes Actuate Information Console, interprets the URI, and communicates with the Actuate BIRT iHub using the Actuate Information Delivery API (IDAPI). The IDAPI manages the request and returns the results to Actuate Information Console and the servlet or page engine. The web server returns the results to the web browser. Then, the web browser displays the results for the user.

Actuate Information Console manages requests as part of a JSP engine within a web or application server. There is no default user interface for the engine. On a Windows system, Actuate Information Console installation places an Actuate Information Console link on the Start menu.

Using proxy servers with Actuate Information Console

When setting up a proxy server with Actuate Information Console, there are steps you must take if your internal application server port is protected by a firewall. In this situation, when the proxy server changes the URL to point to the new

context's port, that port is unavailable due to the firewall. The usual solution is to configure a reverse proxy, but if you are using multiple proxies and a reverse proxy is not practical for your installation, Actuate Information Console can perform the redirection.

To redirect a page without using a reverse proxy, Actuate Information Console forwards the URL to redirect to the `processRedirect.jsp` page and updates the browser's location bar accordingly. This action processes on the client. The browser takes the current URL location and updates the rest of the URI using the redirected URL. You must also set the `ENABLE_CLIENT_SIDE_REDIRECT` configuration parameter to `True` and modify the redirect attributes in the `<context root>/WEB-INF/struts-config.xml` file. The necessary modifications are included in the file. You just need to comment out the lines that have the `redirect` attribute set to `True` and uncomment the lines that forward to the `processRedirect.jsp` page.

For example, the following code is the `struts-config.xml` entry for the login action:

```
<!-- Process a user login -->
<action path="/login" name="loginForm"
    scope="request" input="/iportal/activePortal/private/login.jsp"
    type="com.actuate.activeportal.actions.AcLoginAction"
    validate="false">
    <forward name="loginform"
        path="/iportal/activePortal/private/login.jsp" />
<!--
<forward name="success" path="/iportal/activePortal/private/common
    /processredirect.jsp?redirectPath=/getfolderitems.do" />
-->
<forward name="success" path="/getfolderitems.do"
    redirect="true" />
<forward name="dashboard" path="/dashboard" redirect="true" />
<forward name="ajcLogin" path="/ajclanding.jsp" redirect="true" />
<forward name="landing" path="/landing.jsp" redirect="false" />
</action>
```

By default the forward statement for success points to `getfolderitems.do` with the `redirect` attribute set to `True`. This code instructs the application server to send a redirect with the `getfolderitems.do` URL when the user logs in.

From behind a firewall and proxy, this redirect method fails because the redirect sent by the application server points to the application server port instead of the firewall and proxy port. For success, comment out the line having `redirect="true"`. Uncomment the line that points to `processRedirect.jsp`. The following code shows the updated entry in `struts-config.xml`:

```
<!-- Process a user login -->
<action path="/login" name="loginForm"
    scope="request" input="/iportal/activePortal/private/login.jsp"
    type="com.actuate.activeportal.actions.AcLoginAction"
    validate="false">
```

```

<forward name="loginform"
    path="/iportal/activePortal/private/login.jsp" />
<forward name="success" path="/iportal/activePortal/private/common
    /processredirect.jsp?redirectPath=/getfolderitems.do" />
<!--
<forward name="success" path="/getfolderitems.do"
    redirect="true" />
-->
<forward name="dashboard" path="/dashboard" redirect="true" />
<forward name="ajcLogin" path="/ajclanding.jsp" redirect="true" />
<forward name="landing" path="/landing.jsp" redirect="false" />
</action>

```

This change needs to be made for all the actions in struts-config.xml that send a redirect to the browser.

About Actuate Information Console pages

Actuate Information Console uses JSPs to generate web pages dynamically before sending them to a web browser. These JSPs use custom tags, custom classes, and JavaScript to generate dynamic web page content. The JavaScript, classes, and tags provide access to other pages, JavaBeans, and Java classes. For example, application logic in Actuate Information Console can reside on the web server in a JavaBean.

Web browsers can request a JSP with parameters as a web resource. The first time a web browser requests a page, the page is compiled into a servlet. Servlets are Java programs that run as part of a network service such as a web server. Once a page is compiled, the web server can fulfill subsequent requests quickly, provided that the page source is unchanged since the last request.

The dashboards servlet and JSPs support the dashboards and gadgets interface for Information Console. The dashboard pages reside in <context root>\dashboard\jsp. To provide dashboard access, enable the BIRT 360 license option.

The channels JSPs and custom tags support viewing reports submitted to channels. The channels pages reside in <context root>\iportal\activePortal\private\channels. Users access channels by clicking Channel in the sidebar.

The filesfolders JSPs and custom tags support accessing repository files and folders. These JSPs and custom tags reside in <context root>\iportal\activePortal\private\filesfolders.

The submit request JSPs and custom tags support submitting new jobs. The submit request JSPs reside in <context root>\iportal\activePortal\private\newrequest. For specific information about running jobs using Actuate Information Console, see *Using Information Console*.

The options JSPs and custom tags support managing user option settings. The options pages reside in `<context root>\iportal\activePortal\private\options`.

The viewing JSPs and custom tags support the following functionality, depending on the report type:

- Searching report data
- Using a table of contents to navigate through a report
- Paginating or not paginating a report
- Fetching reports in supported formats

For specific information about viewing reports using Actuate Information Console, see *Using Information Console*.

Use the default pages, customize the pages, or create entirely new pages to deploy your reporting web application.

Working with Actuate Information Console URIs

Actuate Information Console Uniform Resource Identifiers (URIs) convey user requests to the Actuate BIRT iHub System. URIs access functionality including generating and storing reports, managing volume contents, and viewing reports.

About Actuate Information Console URIs

Actuate Information Console URIs consist of the context root and port of the web server where you install and deploy the JSPs or servlets. Actuate Information Console URIs have the following syntax:

```
http://<web server>:<port>/iportal/<path><page>.<type>
[?<parameter=value>{&<parameter=value>}]
```

- `<web server>` is the name of the machine running the application server or servlet engine. You can use `localhost` as a trusted application's machine name if your local machine is running the server.
- `<port>` is the port on which you access the application server or page or servlet engine. The default port for Information Console installed separately is 8700, while the BIRT iHub embedded version uses 8900 by default.
- `iportal` is the default context root for accessing the Actuate Information Console pages.
- `<path>` is the directory containing the page to invoke.
- `<page>` is the name of the page or method.
- `<type>` is `jsp` or `do`.
- `<parameter=value>` specifies the parameters and values that the page requires.

For example, to view the login page, Actuate Information Console uses a URI with the following format:

```
http://<web server>:<port>/iportal  
/login.jsp?TargetPage=<folder/file>
```

- `iportal/login.jsp` is the JSP that provides default login functionality for Information Console.
- `TargetPage` is the `viewframeset.jsp` parameter that specifies the page to direct the user to after the login completes.
- `<folder/file>` is the complete pathname for the file that the client opens after the login completes.

Using a special character in a URI

Actuate Information Console URIs use encoding for characters that a browser can misinterpret. The following example uses hexadecimal encoding in the Information Console URI to display the report, `Newsfeed.rptdesign`, from an Encyclopedia volume:

```
http://localhost:8900/iportal/executereport.do?__requesttype=  
immediate&__executableName=%2fNewsfeed%2erptdesign&__vp=server1
```

You do not have to use hexadecimal encoding in all circumstances. Use the encoding only when the possibility of misinterpreting a character exists. The following unencoded URI displays the same report as the preceding URI:

```
http://localhost:8900/iportal/executereport.do?__requesttype=  
immediate&__executableName=\Newsfeed.rptdesign&__vp=server1
```

Always encode characters that have a specific meaning in a URI when you use them in other ways. Table 1-1 describes the available character substitutions. An ampersand introduces a parameter in a URI, so you must encode an ampersand that appears in a value string. For example, use:

```
&company=AT%26T
```

instead of:

```
&company=AT&T
```

Table 1-1 Encoding sequences for use in URIs

Character	Encoded substitution
ampersand (&)	%26
asterisk (*)	%2a
at (@)	%40
backslash (\)	%5c
colon (:)	%3a

Table 1-1 Encoding sequences for use in URIs

Character	Encoded substitution
comma (,)	%2c
dollar sign (\$)	%24
double quote (")	%22
equal (=)	%3d
exclamation (!)	%21
forward slash (/)	%2f
greater than (>)	%3e
less than (<)	%3c
number sign (#)	%23
percent (%)	%25
period (.)	%2e
plus (+)	%2b
question mark (?)	%3f
semicolon (;)	%3b
space ()	%20
underscore (_)	%5f

If you customize Actuate Information Console by writing code that creates URI parameters, encode the entire parameter value string with the `encode()` method. The `encode()` method is included in `encoder.js`, which is provided in the Actuate Information Console `<context root>/js` directory. The following example encodes the folder name `/Training/Sub Folder` before executing the `getFolderItems` action:

```
<%-- Import the StaticFuncs class. --%>
<%@ page import="com.actuate.reportcast.utils.*" %>

<%
    String url =
        "http://localhost:8900/iportal/getfolderitems.do?folder=" +
        StaticFuncs.encode("/Training/Sub Folder");
    response.sendRedirect(url);
%>
```

The `encode()` method converts the folder parameter value from:

`/Training/Sub Folder`

to:

`%2fTraining%2fSub%20Folder`

About UTF-8 encoding

All communication between Information Console and BIRT iHub uses UTF-8 encoding. UTF-8 encoding is the default encoding that web browsers support. For 8-bit (single-byte) characters, UTF-8 content appears the same as ANSI content. However, if extended characters are used (typically for languages that require large character sets), UTF-8 encodes these characters with two or more bytes.

UTF-8 encoding support is encoded for all Information Console web pages. When customizing these pages or adding customized web pages to an Information Console web application, provide UTF-8 encoding support using the following code:

```
<META
  HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
```

About Actuate Information Console functionality levels

Actuate Information Console provides functionality levels that control which features are available to a user. By default, each user can access all of the functionality level features. To restrict access to features for user groups, the Actuate Information Console administrator can modify functionality levels and add additional levels by editing the configuration file. The standard location for the Actuate Information Console configuration file is <context root>\WEB-INF\functionality-level.config.

Functionality-level.config has several functionality levels mapped to security roles, much like privileges, in comments. Table 1-2 shows the supplied functionality levels and their corresponding security roles.

Table 1-2 Functionality levels mapping to security roles

Functionality level	Security role
Basic	All—default access
Intermediate	Active Portal Intermediate
Advanced	Active Portal Advanced
Administrator	Active Portal Administrator

When uncommenting existing security roles or creating new security roles, make sure that any roles specified in the configuration file also exist in the Encyclopedia volume. Because all users automatically belong to the All security role, all users receive the functionality associated with the Basic or the Open functionality level plus the functionality associated with any other roles they have. When restricting access to features, remove the feature from the Open functionality level or comment out the Open level completely and use the Basic functionality level.

Understanding the provided functionality levels

When the comment tags are removed, the provided functionality levels give the following access. Users with the Basic level can perform the following tasks:

- Access Documents, My Jobs, and Channels.
- Delete their own files.

Basic level users cannot perform any other modifications. The default banner for the Basic level looks like the one in Figure 1-3.

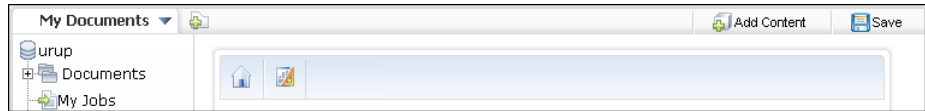


Figure 1-3 Banner menu for a basic level Actuate Information Console user

Users at the Intermediate level have all the Basic level access, and can also perform the following tasks:

- Search documents.
- Create their own job notifications with attachments.
- Subscribe to channels.
- Upload and download files.
- Use the interactive viewer, if this option is licensed.

Users at the Advanced level have all the Intermediate level access, plus they can perform the following tasks:

- Create and delete folders.
- Share files and folders.
- Set job priority.

The default banner for the Intermediate and Advanced levels adds a Search link and looks like the banner in Figure 1-4.

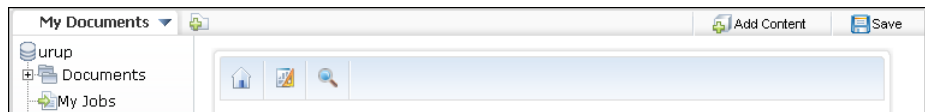


Figure 1-4 Banner menu for advanced level Actuate Information Console user

Users at the Administrator level can perform all Advanced level tasks and can also clone and customize Actuate Information Console skins. The default banner for the Administrator level adds a Customization link, activates the add content function, and looks like the banner in Figure 1-5.

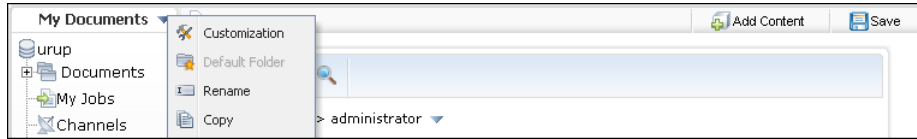


Figure 1-5 Banner menu for an administrator Actuate Information Console user

Use Actuate Management Console to associate the levels with users in the Encyclopedia volume by assigning the appropriate roles to each user.

Customizing functionality levels

Customize or add functionality levels by modifying or creating a level definition in `functionality-level.config`. A functionality level definition consists of five parts:

- **Level name**
The level name must be a unique alphanumeric string, enclosed within `<Name>` and `</Name>` tags.
- **Matching security role**
The name of the security role that corresponds to the functionality level. Both the security level and the functionality level must exist before the functionality level can be assigned to a user. Enclose the role name with `<Role>` and `</Role>` tags.
- **Available features**
Table 1-3 describes the five available features.

Table 1-3 Features for functionality levels

Feature	Description
Channels	Provides access to channels
Customization	Provides access to skin customization
Documents	Provides access to files and folders
Jobs	Allows submitting and accessing jobs
Mobile	Provides access to BIRT mobile viewing
Search	Provides access to the file search facility

Features are specified one per line and are enclosed within `<FeatureID>` and `</FeatureID>` tags. When a feature is omitted from a functionality level, the corresponding side menu or banner item is hidden to anyone assigned that functionality level. For example, the Search feature is not provided in the Basic functionality level, so the Search link does not appear for users with the Basic functionality level.

- Available subfeatures

Subfeatures correspond to actions that you can perform through Actuate Information Console. Most subfeatures are associated with a feature. A subfeature cannot be included in a functionality level if its corresponding feature is not included. The subfeatures are described in Table 1-4.

Table 1-4 Subfeatures for functionality levels

Subfeature	Feature	Description
AddFile	Documents	Permits adding files when the user has the appropriate privileges
AdvancedData	NA	Permits the modifying and synchronizing of data sets in BIRT Studio
CreateFolder	Documents	Permits creating folders when the user has the appropriate privileges
Dashboard BusinessUser	NA	Permits use of dashboards
Dashboard Developer	NA	Permits design and administration of dashboards
DeleteFile	Documents	Permits deleting files when the user has the appropriate privileges
DeleteFolder	Documents	Permits deleting folders when the user has the appropriate privileges
DownloadFile	Documents	Permits downloading files when the user has the appropriate privileges
InteractiveViewing	NA	Permits opening Interactive Viewer
JobPriority	Jobs	Permits setting job priority, up to the user's maximum job priority
SelfNotification WithAttachment	Jobs	Activates e-mail notification for successful jobs
ShareDashboard	NA	Permits sharing dashboards when the user has the appropriate privileges
ShareFile	Documents	Permits sharing files when the user has the appropriate privileges
SubscribeChannel	Channels	Permits subscribing to channels

Subfeatures are specified one per line, enclosed within <SubfeatureID> and </SubfeatureID> tags.

The following code shows a sample functionality level entry:

```
<Level>
  <Name>ViewAndSearch</Name>
  <Role>All</Role>
  <FeatureID>Jobs</FeatureID>
  <FeatureID>Documents</FeatureID>
  <FeatureID>Search</FeatureID>
  <SubfeatureID>ShareFile</SubfeatureID>
  <SubfeatureID>DeleteFile</SubfeatureID>
</Level>
```

The level is named ViewAndSearch and is available to all security roles. Users with ViewAndSearch functionality can run jobs, access documents, and search for files. In addition, they can share and delete their own files.

Customizing functionality level features

Customize functionality level features by modifying the action they perform and the graphic image they use. Features are defined in the functionality-level.config file. A feature definition consists of up to five parts:

- **Feature ID**

This is the feature name and must be a unique alphanumeric string, enclosed within <ID> and </ID> tags. This value is used as the feature name in functionality level definitions. Do not change this value, because the IDs are used in the Actuate Information Console code to identify the features.

- **Label key**

This key is used in the Actuate Information Console resource files. These files have names of the format, ActivePortalResources_<locale>.properties. The files are located in <context root>\WEB-INF\lib\resources.jar. If this file does not contain a resource file for a locale, the resource file, ActivePortalResources.properties, for the default locale, en_US, is used. The key provides for proper translation in the resource file so that the hyperlink text for the feature is displayed using the current locale. Keys are enclosed within <Labelkey> and </Labelkey> tags. Do not change the key values or the resource string substitution fails.

- **Link**

This link is the target URI of the label key hyperlink, which is typically to the page that corresponds to the feature. Table 1-5 shows the targets for each feature. Links are enclosed within <Link> and </Link> tags. Change the link target for the feature by replacing the default page or action name.

Table 1-5 Actuate Information Console targets for features

Feature	Actuate Information Console target
Documents	\getfolderitems.do
Jobs	\selectjobs.do
Channels	\selectchannels.do
Search	\searchfiles.do
Customization	\customize.do

■ Large icon and Small icon

These optional icons are displayed together with the link, depending on the skin. For example, the Classic skin displays the large icons, the Treeview skin uses the small icons, and the Tabbed skin does not use these icons at all. Table 1-6 shows features and their icons. Large icons are 32 pixels square. Their file names are relative to the context root and are enclosed within <LargeIcon> and </LargeIcon> tags. Small icons are 16 pixels square. Their file names are relative to the context root and are enclosed within <SmallIcon> and </SmallIcon> tags. Replace these file names with the names of your own icons to customize your skin's appearance.

Table 1-6 Icons for features

Feature	SmallIcon	LargeIcon
Documents	\images\filesfoldersicon16x16.gif	\images\filesfoldersicon.gif
Jobs	\images\requestsicon16x16.gif	\images\requestsicon.gif
Channels	\images\channelsicon16x16.gif	\images\channelsicon.gif

The following example shows a sample definition for the Channels feature. This example specifies custom large and small icons. The Classic and Treeview skins, and any skins cloned from them, use these new images for the channel icon.

```
<Feature>
  <ID>Channels</ID>
  <Labelkey>SBAR_CHANNELS</Labelkey>
  <Link>/selectchannels.do</Link>
  <SmallIcon>/images/customIcon16x16.gif</SmallIcon>
  <LargeIcon>/images/customIcon32x32.gif</LargeIcon>
</Feature>
```

Preserving functionality levels and features

The `functionality-levels.config` file is overwritten during upgrade installations. This change ensures that new levels, features, and subfeatures are available to you with your new Actuate Information Console installation. If you have modified your existing `functionality-level.config` file, make a backup of the changes before the upgrade. Use the backed-up file to access your changes and merge them into the new `functionality-level.config` file.

Creating a custom Information Console web application

This chapter contains the following topics:

- Information Console web application structure and contents
- Configuring a custom Information Console web application
- Customizing an Information Console web application
- Modifying global style elements

Information Console web application structure and contents

Information Console generates web pages using a set of default JSPs. Actuate Information Console JSPs use cascading style sheets, JavaScript, and custom tags to generate dynamic web page content. The JavaScript and tags provide access to other JSPs, JavaBeans, and Java classes.

The Information Console web application organizes these inter-operating components into a Model-View-Controller (MVC) architecture. To operate a web application, the MVC components perform the following functions:

- Model contains the logic for sending requests to and processing responses from the repository. This component is the data model for Information Console.
- View contains the pages that display data prepared by actions. This component is the presentation portion of Information Console.
- Controller contains the servlets that implement actions. This component is the program control logic for Information Console and manages actions initiated from the browser.

The controller maps actions, designated by URLs with the .do extension, to an `actionServlet`. The `actionServlet` is configured with action paths specified in `<Actuate home> \iPortal\iportal\WEB-INF\struts-config.xml`.

Typically, an action path leads to a JSP with parameters as a web resource. Actuate Information Console file and directory names are case-sensitive. The first time you use a JSP, your web server compiles it into a servlet. Servlets are compiled Java programs or JSPs that run as part of a network service such as a web server. After compiling a JSP into a servlet, a web server can fulfill subsequent requests quickly, provided that the JSP source does not change between requests.

Users make requests to view the contents of a repository, run and view reports, and so on. Each JSP processes any URL parameters by passing them to JSP tags, including Actuate custom tags or your own custom tags.

You specify the user's Actuate BIRT iHub System and Encyclopedia volume as URL parameters. To specify the locale and time zone to which to connect, use parameter values in an Actuate Information Console request within a URL or by specifying the desired values in the login form. For example, the following URL specifies the `en_US` locale for U.S. English, and the Pacific standard time for the `timezone` parameter:

```
http://localhost:8900/iportal/login.do?locale=en_US&timezone=PST
```


Understanding Information Console directory structure

The Java Server Pages (JSPs) that implement Actuate Information Console URIs are grouped by method into directories under the context root. The context root is the home directory in which an Actuate Information Console web application resides. The default context root for the embedded Information Console for iHub on Windows systems is <Actuate home>\iHub2\servletcontainer\iportal and on UNIX and Linux systems is <Actuate home>/iHub2/servletcontainer/iportal. The default context root for a separate Information Console installation on Windows systems is <Actuate home>\iPortal2\iportal and on UNIX and Linux systems is <Actuate home>/iPortal2/iportal. The Information Console context root name in the web or application server’s configuration file is iportal. Figure 2-1 shows the Information Console directory structure.

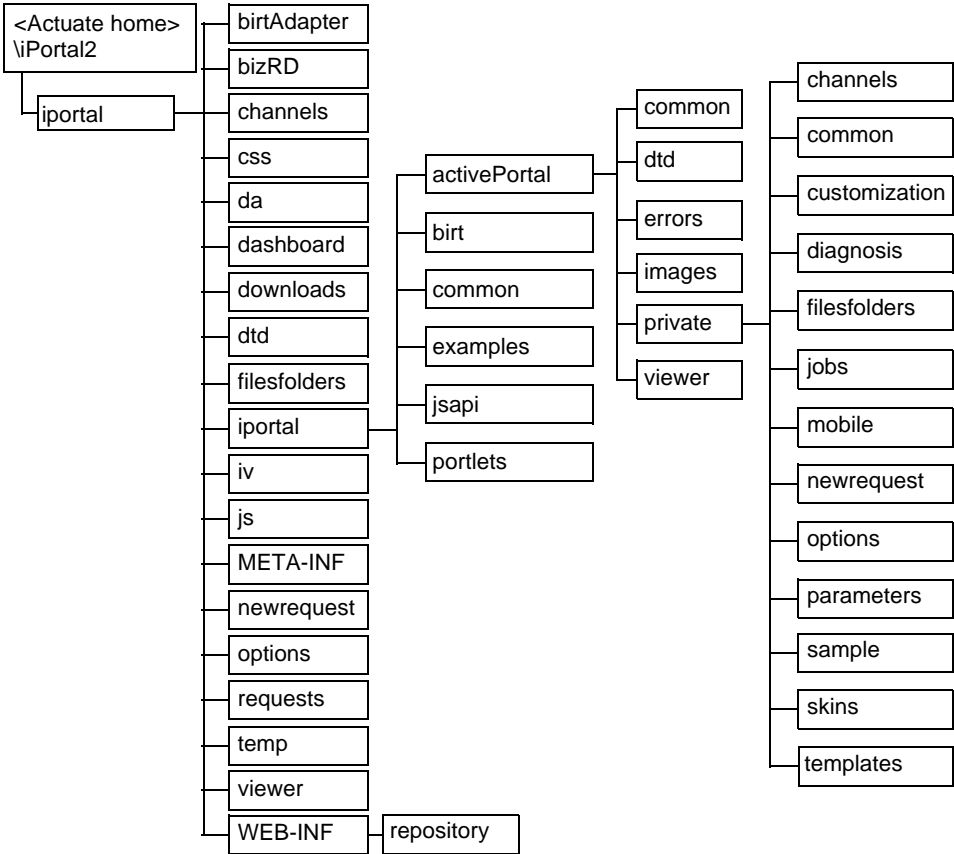


Figure 2-1 Actuate Information Console directory structure

Actuate Information Console URIs convey user requests to Actuate BIRT iHub.

Pages supporting folder navigation and document viewing reside in the <context root>\iportal\activePortal directory. In this directory, pages supporting report viewing reside in the viewer directory, pages serving as templates for other pages reside in the templates directory, and so on. Some directory names exist in the iportal directory and also in the <context root>\iportal\activeportal\private subdirectory. Customize the JSPs in the private subdirectory. The directory of the same name in the iportal directory exists only for backward compatibility. Table 2-1 lists and describes the general iHub2\servletcontainer\iportal or iPortal\iportal directories.

Table 2-1 <Context root> directories

Directory	Contents
This directory	landing.jsp, the default page for accessing all Information Console functionality.
birtAdapter	BIRT Viewer integration files.
bizRD	Pages that support BIRT Studio.
channels	Pages that support channels.
css	Actuate Information Console cascading style sheet (.css) files.
da	BIRT Data Analyzer support files.
dashboard	Dashboard support files.
downloads	Downloaded files.
dtd	Document type definitions.
filesfolders	Pages that support working with files and folders.
images	Information Console user interface images and icons.
iportal	The Information Console application.
iv	Pages that support Interactive Viewer.
js	JavaScript files that control specific web page elements such as search, toolbar, and table of contents.
logs	Administrative and SOAP fault log files.
META-INF	The Information Console manifest file. iHub embedded InfoConsole directory.
newrequest	Pages that support new requests, such as parameter processing, scheduling, and job status pages.
options	Options-specific pages, such as channels, notification, and options update pages.

Table 2-1 <Context root> directories

Directory	Contents
requests	Pages in this directory provide backward compatibility for custom web applications referencing these pages by URL. Use the action paths and the private\jobs directory for new customization projects.
temp	Working directory for transient content.
viewer	Pages that support report viewing.
WEB-INF	Files that manage session information such as current user login, roles, and volume.

Table 2-2 lists and describes the iportal directories.

Table 2-2 <Context root>/iportal directories

Directory	Contents
activePortal	Pages that support login and authentication and directories for the remaining pages for folder navigation and document usage
birt	Libraries that support BIRT reports, BIRT Studio, and Interactive Viewer and pages that support BIRT reports
common	Common elements included in all reporting web pages, such as banner and side menu elements
jsapi	The Java Report Engine Manager

Table 2-3 lists and describes the <context root>\iportal\activePortal directories.

Table 2-3 <Context root>/iportal/activePortal directories

Directory	Contents
This directory	Pages that support login and authentication and directories for the remaining folder and document pages for the Information Console application.
common	Common elements included in all reporting web pages, such as banner and side menu elements.
dtd	Document type definitions.
errors	Error pages.
images	Images for reporting web pages, such as buttons, icons, lines, and arrows.

(continues)

Table 2-3 <Context root>/portal/activePortal directories (continued)

Directory	Contents
private	Most Information Console folders and documents web pages. Users cannot directly access pages in this directory using URLs. These pages are customizable.
private \channels	Pages that support channels.
private \common	Common elements included in all reporting web pages, such as banner and side menu elements.
private \customization	Pages that support customization of skins.
private \diagnosis	Self-diagnostic utility page.
private \filesfolders	Pages that support working with files and folders.
private\jobs	Pages that support requests such as completed requests, successful submission, and details pages by redirecting.
private\mobile	Pages that support BIRT Mobile subscriptions.
private \newrequest	Pages that support new requests, such as parameter processing, scheduling, and job status pages.
private\options	Options-specific pages, such as channels, notification, and options update pages.
private \parameters	Pages that support table parameters.
private\sample	Example custom requester page.
private\skins	Skins definitions.
private \templates	Jakarta Struts template pages that simplify customization by handling common web page structure and functionality for many pages.
viewer	Pages that support report viewing.

Actuate recommends that you group Information Console applications in the home directory of an Actuate distribution to make them easier to locate. Place the context root in whatever location your application requires. To ensure that the JSP engine locates your Information Console application's context root, add its location to your JSP engine's configuration file as a context root path.

Building a custom Information Console context root

Application servers route requests from the user's browser to the configured Information Console web content in a context root. A JSP engine specifies the path for the Information Console context root in a platform-specific configuration file. For example, the Tomcat engine specifies context roots in the `/etc/tomcat/server.xml` file on a UNIX or Linux system and `C:\Program Files (x86)\Apache Software Foundation\Tomcat 6.0\conf\server.xml` file on a Windows system. Other application servers and servlet engines use an analogous file.

You can configure multiple Actuate Information Console context roots on a single server. Each context root can contain a web reporting application that uses a different design. For example, you can create different web reporting applications for particular language groups or departments. The following example is the definition for the default Actuate Information Console context root, `iportal`, from a Tomcat `server.xml` file on a Windows system:

```
<Context
  path="/iportal"
  docBase="C:\Program Files (x86)\Actuate\iPortal2\iportal"
  debug="0"/>
```

The following example is the definition for the default Actuate Information Console context root, `iportal`, from a Tomcat `server.xml` file on a UNIX-based system:

```
<Context
  path="/iportal"
  docBase="/home/user/iPortal/iportal"
  debug="0"/>
```

Actuate Information Console's embedded servlet engine uses an automatic mechanism to discover new web applications. This server provides a quick and convenient environment in which to test your custom Information Console application before deploying to your main application server. To test a custom Information Console application on the embedded servlet engine, you create the context root directory structure in `<Actuate home>iPortal2`, then restart the Actuate 11 Apache Tomcat for Information Console service and clear your browser cache.

How to create a new context root

The following example creates a custom web application for MyCorp's Marketing Communications group. The Marketing Communications users use the following URI prefix to access their custom application:

```
http://MyCorp:8700/marcom
```

For example, to access the application's login page, they choose a web page hyperlink with the following URI:

```
http://MyCorp:8700/marcom/login.do
```

- 1 Install Information Console separately. Information Console installed separately is portable but Information Console embedded in BIRT iHub is not.
- 2 Make a copy of the Actuate Information Console directory structure and give the copy a name related to the context root name.

For example, for the default installation on a Windows machine, copy the directory C:\Program Files\Actuate\iPortal2\iportal, paste it into C:\Program Files\Actuate\iPortal2 and rename it marcom.

For the default installation on a UNIX-based machine, copy the directory \$HOME/iPortal2/iportal, paste it into \$HOME/iPortal2/ and rename it marcom.

- 3 If you are using a server other than Information Console's embedded servlet engine, add your definition to the JSP engine's configuration file. For example, for Tomcat installed on a Windows machine, add the context root, marcom, to the <Information Console Directory>\conf\server.xml file as follows:

```
<Context
  path="/marcom"
  docBase="C:\Program Files\Actuate\iPortal2\marcom"
  debug="0"/>
```

For Tomcat installed on a UNIX-based machine, add the context root, marcom, to the <Information Console Directory>/conf/server.xml file as follows:

```
<Context
  path="/marcom"
  docBase="/home/user/iPortal2/marcom"
  debug="0"/>
```

- 4 Restart the application server or JSP engine. For example, to restart Information Console's embedded servlet engine on a Windows XP system, perform the following steps:
 - 1 From the Windows Start menu, choose All Programs>Administrative Tools>Services.
 - 2 On Services, select Actuate 11 Apache Tomcat for Information Console service.
 - 3 From the menu, choose Action>Restart.
 - 4 Close Services.

To restart Information Console's embedded servlet engine on a UNIX-based system, perform the following steps:

- 1 Open a console window.
- 2 Type the following commands:

```
sh $HOME/iPortal2/iportal/actuate_http_service/bin/shutdown.sh
sh $HOME/iPortal2/iportal/actuate_http_service/bin/startup.sh
```

After you stop and restart the server, the Marketing Communications users can access the Actuate Information Console web application called marcom. The application looks like the default Actuate Information Console application because you have not customized its appearance.

Activating a new or custom web application

To activate the changes you make in the Information Console configuration files, content pages, or by creating a new context root, you must restart the web server that runs Information Console and clear the browser cache. For the default Information Console installation, you restart the Actuate 11 Apache Tomcat for Information Console service.

How to restart the Actuate 11 Apache Tomcat for Information Console service on a Windows XP system

- 1 From the Windows Start menu, choose All Programs→Administrative Tools→Services.
- 2 On Services, select Actuate 11 Apache Tomcat for Information Console service.
- 3 From the menu, choose Action→Restart.
- 4 Close Services.

How to clear the browser cache for Microsoft Internet Explorer 7

- 1 Open Microsoft Internet Explorer.
- 2 Choose Tools→Delete Browsing History
- 3 On Delete Browsing History, choose Delete All. Then choose Yes.
- 4 Close Microsoft Internet Explorer.

Configuring a custom Information Console web application

Information Console's configuration determines many of its essential methods. Configuring your web application customizes how it operates internally, and affects the user's experience.

Customize specific pages and operations using the Actuate Information Console web pages, as described in "Customizing an Information Console web application," later in this chapter.

Perform cosmetic customization tasks using the Actuate Information Console skins and style sheets, as described in "Modifying global style elements," later in this chapter.

Customizing Information Console configuration

Set configuration parameters for the Information Console application to tune performance and to control service and application execution. For example, you can perform the following tasks using configuration parameters:

- Setting the default locale
- Controlling the Message Distribution service load balancing
- Specifying the default Encyclopedia volume and server

You configure the Information Console application by changing configuration file contents, such as `web.xml`. To understand the common configuration files and how each of their entries affect Information Console, see Chapter 3, “Actuate Information Console configuration.”

The following section describes the customization procedure using the text editor.

How to customize Information Console configuration parameters

Use the following procedure to customize configuration parameters for Information Console. In this procedure, it is assumed that `<context root>\WEB-INF\web.xml` is the configuration file.

- 1 Make a backup copy of `web.xml`.
- 2 Using a text editor that supports UTF-8 encoding, edit `web.xml` to change parameter values. Parameter definitions use the following format:

```
<param-name><keyword></param-name>
<param-value><value></param-value>
```

- `<keyword>` is the name of the parameter.
- `<value>` is the parameter value.

Do not enclose the keyword and value within quotes, and use no spaces between `<param-name>`, the keyword or value, and `</param-name>`. For example, the definition for the default locale parameter is:

```
<param-name>DEFAULT_LOCALE</param-name>
<param-value>en_US</param-value>
```

- 3 Save `web.xml`.
- 4 Restart the application server or servlet engine that runs Information Console and clear your browser cache.

Setting the default locale

The default locale and time zone for Information Console are set when you install it. To change the default settings, you modify the values of the `DEFAULT_LOCALE` and `DEFAULT_TIMEZONE` configuration parameters.

How to set a default Information Console locale and time zone

- 1 Using a UTF-8 compliant code editor, open the web.xml configuration file.
- 2 Navigate to the lines that define DEFAULT_LOCALE, similar to the following code:

```
<param-name>DEFAULT_LOCALE</param-name>
<param-value>en_US</param-value>
```

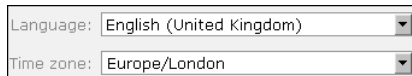
Change the current locale id, en_US in the above example, to the desired locale id in param-value. Valid locale id strings are listed in <context root>\WEB-INF\localemap.xml.

- 3 Navigate to the lines that define DEFAULT_TIMEZONE, similar to the following code:

```
<param-name>DEFAULT_TIMEZONE</param-name>
<param-value>America/Los_Angeles</param-value>
```

Change the current time zone id, Pacific Standard Time in the above example, to the desired default time-zone in param-value. Valid time zone id strings are listed in <context root>\WEB-INF\TimeZones.xml.

- 4 Save web.xml.
- 5 Restart the application server or servlet engine that runs Information Console and clear your browser cache.
- 6 Open the Information Console web application. The login page for the custom application appears. A login page with default locale set to en_GB, and the default time zone set to Europe/London, appears as shown in Figure 2-2.



Language:	English (United Kingdom)
Time zone:	Europe/London

Figure 2-2 The login page for the custom application

Controlling the Message Distribution service load balancing

The default load balancing for Information Console are set to when you install it. To change the default settings, you modify the values of the MDS_ENABLED and MDS_REFRESH_FREQUENCY_SECONDS configuration parameters.

If you are using third-party load balancing, you will need to refer to their documentation to configure load balancing. See “Understanding Actuate Information Console load balancing” in Chapter 1, “Introducing Actuate Information Console.”

How to enable the Message Distribution service

The Message Distribution service (MDS) is enabled by default. This procedure assumes it has been disabled.

- 1 Using a UTF-8 compliant code editor, open the web.xml configuration file.
- 2 Navigate to the lines that define MDS_ENABLED, similar to the following code:

```
<param-name>MDS_ENABLED</param-name>  
<param-value>>false</param-value>
```

Change the current value, if it is false, to true.

- 3 Navigate to the lines that define MDS_REFRESH_FREQUENCY_SECONDS, similar to the following code:

```
<param-name>MDS_REFRESH_FREQUENCY_SECONDS</param-name>  
<param-value>0</param-value>
```

Change the current refresh frequency in seconds, 0 in the above example, to the desired number of seconds so that MDS will attempt to discover new nodes added to the cluster or remove nodes dropped from the cluster.

- 4 Save web.xml.
- 5 Restart the application server or servlet engine that runs Information Console and clear your browser cache.

Specifying the default Encyclopedia volume and server

The default Encyclopedia volume and server is set when you install Information Console to the local web service and machine name. To use a different Encyclopedia volume and server by default or hide this information in the URL using a volume profile name, you add a profile to the VolumeProfiles.xml configuration file.

How to specify the default Encyclopedia volume and server

- 1 Using a UTF-8 compliant code editor such as JCreator, open the VolumeProfile.xml configuration file
- 2 Navigate to the lines that define the default Profile, similar to the following code:

```
<Profile>  
  <Default>true</Default>  
  <ProfileName>LocalMachine</ProfileName>  
  <RepositoryType>enterprise</RepositoryType>  
  <ServerUrl>http://LocalMachine:8000</ServerUrl>  
  <Volume>LocalMachine</Volume>  
</Profile>
```

Navigate to the line that defines Default, and change the value from true to false.

- 3 Create a copy of the entire LocalMachine profile immediately below the LocalMachine profile's </Profile> tag and before the </VolumeProfiles> tag.

- 4 Change the values of your copied profile to the new default Encyclopedia volume and server, similar to the following code:

```
<Profile>
  <Default>true</Default>
  <ProfileName>NewServer</ProfileName>
  <RepositoryType>enterprise</RepositoryType>
  <ServerUrl>http://NewServer:8000</ServerUrl>
  <Volume>NewServer</Volume>
  <DashboardTemplatePath></DashboardTemplatePath>
</Profile>
```

- The value of Default is true, indicating that the profile is the default server profile. Set only one profile Default to true in VolumeProfile.xml, the others must be set to false.
 - The value of ProfileName is a unique name for the server profile.
 - The value of ServerUrl is the URL for the new iHub service to contact by default.
 - The value of Volume is the name of the Encyclopedia volume to access by default.
 - The value of DashboardTemplatePath is an optional repository path for a dashboard file that Information Console loads by default when creating new dashboards.
- 5 Save VolumeProfile.xml. Close the code editor.
 - 6 Restart the application server or servlet engine that runs Information Console and clear your browser cache.
 - 7 Open the Information Console web application. The login page for the custom application appears. The URL will contain the default volume profile information in the VolumeProfile parameter, similar to the following:

```
http://localhost:8900/iportal/login.jsp?
  &__vp=NewServer&targetPage=/iportal/getfolderitems.do
```

Modifying text and messages

Actuate Information Console provides text and messages and also passes Actuate BIRT iHub messages to the user. You can customize both Actuate BIRT iHub and Actuate Information Console messages and text. Actuate has created the Actuate Information Console software and resource files in multiple languages. If you need to change the text and messages to translate your Actuate Information Console web application to another language, contact Actuate Corporation.

Customizing Information Console text and messages

Actuate Information Console uses text and messages to communicate with the user. Customize the text of a label to prompt your user with the phrasing that your application needs by changing configuration files in one or more of the files in `resources.jar`, located in `<context root>\WEB-INF\lib\`. For example, the default title of the landing page displayed in the title bar and tab text of your web browser is Actuate Information Console, as shown in Figure 2-3.

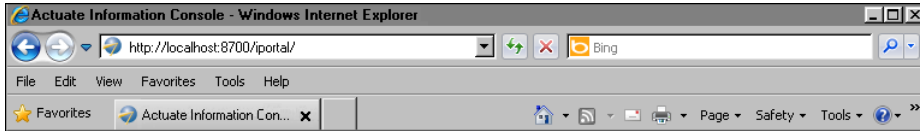


Figure 2-3 Default title bar text of the Information Console landing page

To change this title, change the value of the `TITLE_LANDING_PAGE` parameter in `com\actuate\portal\common\bundle\messages.properties` file compressed in the `<context root>\WEB-INF\lib\com.actuate.resources.jar`. By editing `TITLE_LANDING_PAGE`, you can customize the marcom web site by replacing the default title with Marcom Information Console, as shown in Figure 2-4.

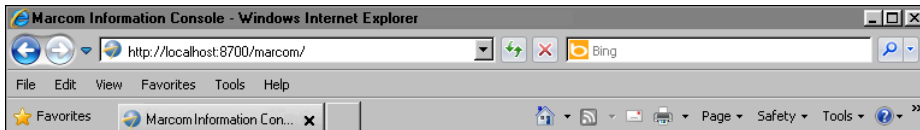


Figure 2-4 Custom title bar text of the Information Console landing page

You can find the method of a particular line of text in the Information Console web application by searching for the relevant message key in the JSPs and examining the related code. To customize a message in other parts of Information Console, you edit the appropriate properties file compressed in `resources.jar`. Table 2-4 lists the properties files that provide messages and text to particular Information Console page categories.

Information Console inserts additional text using variables. When customizing messages and text, keep the original variables in your text or message, if possible. Variables appear in text and messages in the form `{n}` where `n` is a whole number, beginning with 0.

For example, if a user `mlee` tries to subscribe to a channel but has no available channels other than the user's personal channel, Information Console displays the `MSGT_NO_CHANNELS` message and its variable from `com\actuate\activeportal\resources\ActivePortalResources.properties`:

There are no channels available for subscription by `{0}`.

in the following form:

There are no channels available for subscription by `mlee`.

How to customize Actuate Information Console text and messages on a Windows system

Use the location of your Information Console installation if it differs from the location used in this example.

- 1 Extract the contents of <context root>\WEB-INF\lib\resources.jar into a temporary directory.

- 1 Open a command window.

- 2 Back up your resources file:

```
cd "C:\Program Files\Actuate\iPortal2\iportal\WEB-INF\lib"
copy com.actuate.resources.jar
      com.actuate.resources.jar.original
```

- 3 Extract the resource file's contents:

```
mkdir C:\ap
cd C:\ap
jar -xf "C:\Program Files\Actuate\iPortal2\iportal\WEB-INF
      \lib\com.actuate.resources.jar"
```

- 4 Leave the command window open.

- 2 Navigate to com\actuate\activeportal\resources and make a backup copy of ActivePortalResources.properties:

```
cd com\actuate\activeportal\resources
copy ActivePortalResources.properties
      ActivePortalResourcesOrig.properties
```

- 3 In a text editor that supports UTF-8 encoding, edit C:\ap\com\actuate\reportcast\resources\ActivePortalResources.properties to add your custom error messages in the following format:

```
<Errorcode>=Example of a message with no variables.
<Errorcode>=Example of a message with a variable {0}.
<Errorcode>=Message with three variables {0}, {1} and {2}.
```

where <Errorcode> is the Actuate error number or constant of the message being customized.

- 4 Save and close the file.
- 5 Rebuild the resources.jar file with your customized ActivePortalResources.properties file:

```
cd C:\ap
jar -cf com.actuate.resources.jar *
move resources.jar "C:\Program Files\Actuate\iPortal2
      \iportal\WEB-INF\lib\com.actuate.resources.jar"
```

How to customize Actuate Information Console text and messages on a Linux system

Use the location of your Information Console installation if it differs from the location used in this example.

- 1 Extract the contents of resources.jar into a temporary directory:

- 1 Back up your resources file:

```
cd /usr/local/Actuate/iPortal2/iportal/WEB-INF/lib
cp com.actuate.resources.jar
   com.actuate.resources.jar.original
```

- 2 Extract the resource file's contents:

```
mkdir ap
cd ap
jar -xf /usr/local/Actuate/iPortal2/iportal/WEB-INF/lib
      /com.actuate.resources.jar
```

- 2 Navigate to com/actuate/activeportal/resources and make a backup copy of ActivePortalResources.properties:

```
cd com/actuate/activeportal/resources
cp ActivePortalResources.properties
   ActivePortalResourcesOrig.properties
```

- 3 In a text editor that supports UTF-8 encoding, edit ap/com/actuate/reportcast/resources/ActivePortalResources.properties to add your custom error messages in the following format:

```
<Errorcode>=Example of a message with no variables.
<Errorcode>=Example of a message with a variable {0}.
<Errorcode>=Message with three variables {0}, {1} and {2}.
```

where <Errorcode> is the Actuate error number or constant of the message being customized.

- 4 Save and close the file.
- 5 Rebuild the resources.jar file with your customized ActivePortalResources.properties file:

```
jar -cf resources.jar *
mv resources.jar /usr/local/Actuate/iPortal2/iportal
  /WEB-INF/lib/com.actuate.resources.jar
```

Customizing Actuate BIRT iHub error messages

Actuate Information Console uses SOAP messages to communicate with the Actuate BIRT iHub. You can customize the message text of an Actuate BIRT iHub error message before Information Console displays it to the user. For example, the following URL attempts to schedule a job for a report that is not in the repository:

```
http://localhost:8700/iportal/submitjob.do
?requesttype=scheduled&executableName=BadFileName.x
```

Information Console retrieves an iHub error message, as shown in Figure 2-5.

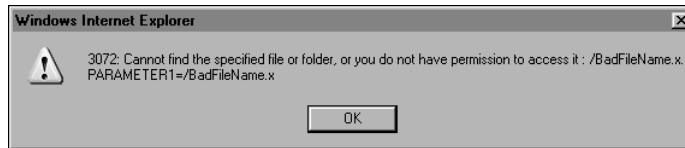


Figure 2-5 iHub error message for a missing file

To customize a message, you edit `ErrorMessages.properties`, following the procedures described later in this section. This file contains customized error messages. For a full list of all BIRT iHub error messages, see `<context root>\WEB-INF\ErrorMessages.txt`. This file contains the error code, error level, and the English text of every message. When you customize `ErrorMessages.properties`, use the error code for the message from `ErrorMessages.txt`.

Information Console inserts context-specific text to an error message using variables. When changing message text, maintain the original variables in your new message, if possible. For the best results, follow the format of the original message exactly to maintain the number and order of the variables. Variables appear in message text as `{n}` where `n` is a whole number, beginning with 0.

For example, the URL for a missing file produces error 3072, and you can change the entry for error 3072 to something similar to the following:

```
3072 = {0} is a bad file name or the file does not exist.
```

Using the erroneous URL above with this custom message results in a new message, as shown in Figure 2-6.



Figure 2-6 Custom iHub error message for a missing file

How to customize Actuate BIRT iHub error messages on a Windows system

Use the location of your own Actuate Information Console installation if it differs from the location used in this example.

- 1 Extract the contents of `<context root>\WEB-INF\lib\resources.jar` into a temporary directory.
 - 1 Open a command window.

2 Back up your resources file:

```
cd "C:\Program Files\Actuate\iPortal2\iportal\WEB-INF\lib"
copy com.actuate.resources.jar
    com.actuate.resources.jar.original
```

3 Extract the resource file's contents:

```
mkdir C:\ap
cd C:\ap
jar -xf "C:\Program Files\Actuate\iPortal2
    \iportal\WEB-INF\lib\com.actuate.resources.jar"
```

4 Leave the command window open.

2 Navigate to com\actuate\reportcast\resources and make a backup copy of ErrorMessage.properties:

```
cd com\actuate\reportcast\resources
copy ErrorMessage.properties ErrorMessageOrig.properties
```

3 In a text editor that supports UTF-8 encoding, edit C:\ap\com\actuate\reportcast\resources\ErrorMessage.properties to add your custom error messages in the following format:

```
<Errorcode>=Example of a message with no variables.
<Errorcode>=Example of a message with a variable {0}.
<Errorcode>=Message with three variables {0}, {1} and {2}.
```

where <Errorcode> is the Actuate error number or constant of the message being customized.

4 Save and close the file.

5 Rebuild the resources.jar file with your customized ErrorMessage.properties file:

```
jar -cf resources.jar *
move resources.jar "C:\Program Files\Actuate\iPortal2
    \iportal\WEB-INF\lib\com.actuate.resources.jar"
```

How to customize Actuate BIRT iHub error messages on a UNIX or Linux system

Use the location of your Information Console installation if it differs from the location used in this example.

1 Extract the contents of resources.jar into a temporary directory:

1 Back up your resources file:

```
cd /usr/local/Actuate/iPortal2/iportal/WEB-INF/lib
cp com.actuate.resources.jar
    com.actuate.resources.jar.original
```


2 Extract the resource file's contents:

```
mkdir ap
cd ap
jar -xf /usr/local/Actuate/iPortal2/iportal/WEB-INF/lib
/com.actuate.resources.jar
```

2 Navigate to `com/actuate/activeportal/resources` and make a backup copy of `ErrorMessages.properties`:

```
cd com/actuate/activeportal/resources
cp ErrorMessages.properties ErrorMessagesOrig.properties
```

3 In a text editor that supports UTF-8 encoding, edit `ap/com/actuate/reportcast/resources/ErrorMessages.properties` to add your custom error messages in the following format:

```
<Errorcode>=Example of a message with no variables.
<Errorcode>=Example of a message with a variable {0}.
<Errorcode>=Message with three variables {0}, {1} and {2}.
```

where `<Errorcode>` is the Actuate error number or constant of the message being customized.

4 Save and close the file.

5 Rebuild the `resources.jar` file with your customized `ErrorMessages.properties` file:

```
jar -cf resources.jar *
mv resources.jar /usr/local/Actuate/iPortal2/iportal/WEB-INF
/lib/com.actuate.resources.jar
```

Customizing an Information Console web application

To perform most cosmetic customization tasks, use the Actuate Information Console skin manager. The skin manager supports using skins to change typically customized images, colors, and fonts in Actuate Information Console web pages. You also can customize aspects of Information Console that are not supported by the skin manager by modifying the Information Console files manually.

Actuate Information Console supports customization of the landing page, `<context root>\landing.jsp`, and the appearance of the pages in My Documents, BIRT Studio, and the interactive viewer for BIRT reports.

You use knowledge of the following standard languages and frameworks to customize an Information Console web application manually:

- **Cascading style sheet (.css) files**
CSS files define fonts, colors, and other visual design attributes of an Information Console web application. For information about modifying style sheets, see “Modifying global style elements,” later in this chapter.
- **Hypertext markup language (HTML)**
HTML handles links and the presentation of text and graphics in web pages. Information Console incorporates HTML code in its JavaServer pages.
- **Jakarta Struts Framework**
Jakarta Struts Framework is an open source framework for building web applications. Based on standard technologies, Struts enables the Information Console Model-View-Controller design. For more information about Struts, access the following URL:

`http://jakarta.apache.org/struts`
- **Java**
Information Console uses Java classes to provide functionality. You can create your own Java classes for your custom web application. For more information on the Information Console Java classes, see Chapter 8, “Actuate Information Console JavaBeans.”
- **JavaScript**
JavaScript is an interpreted, object-oriented language that facilitates embedding executable content in web pages. It provides strong tools for interacting with web browsers.
- **JavaServer Pages**
The JavaServer Pages (JSP) extension of the Java Servlet API facilitates the separation of page design from business logic. JSPs are a platform-independent solution. Information Console web pages are defined primarily by JSPs. For more information about the Actuate JavaServer Pages, see Chapter 4, “Actuate Information Console URIs.”

Actuate recommends that you use the skin manager to customize as much as possible and then handle any remaining customization tasks manually.

Modifying the landing page

To modify the appearance of the landing page, use custom styles as described later in this chapter. The landing page uses the same cascading style sheets files as the other Actuate Information Console JSPs. Figure 2-7 shows some of the classes that define various elements of the landing page. Where possible, modify these styles by using the customization pages for skins.

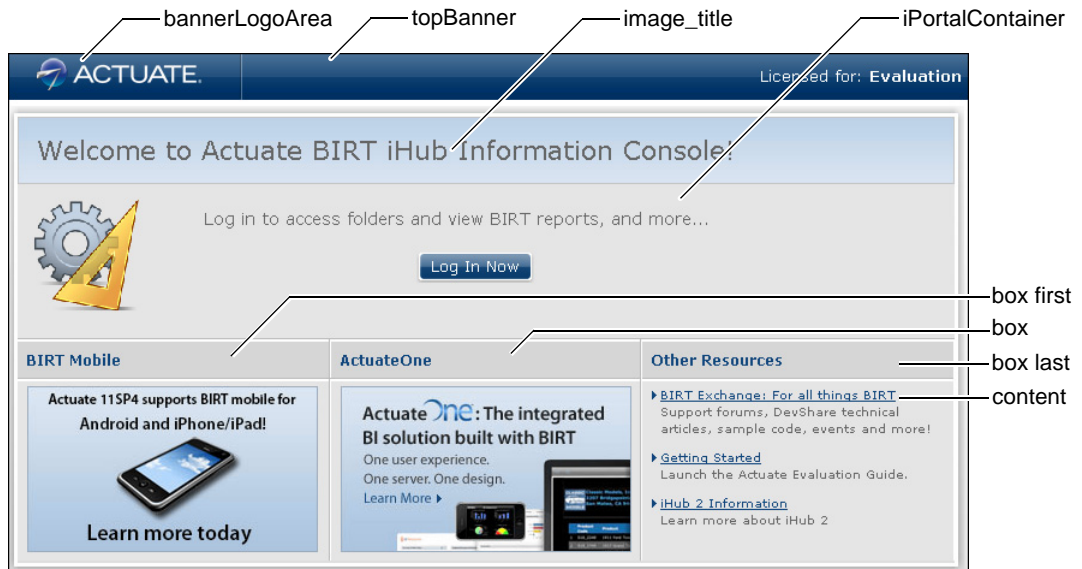


Figure 2-7 Classes used on the default landing page

Use the contents of the `<body>` element in `landing.jsp` to customize the branding images, the welcome text in the banner, and the list of links that appears at the left of the default landing page. The code comments indicate which portion of the page can be modified. For example, the code after the comment `<%-- THE UPPER SECTION --%>` refers to the banner area at the top of the page.

Viewing modifications to a custom web application

After making changes to your Information Console web application, you need to view the changes. Caching in the browser or your application server can interfere with seeing the changes you made. After changing an Information Console application, complete these general tasks in order:

- Save any files involved in the change.
- Refresh the browser page.
- If you do not see changes you made in a JSP or XML file, complete the following tasks in order:
 - Shut down the JSP engine.
 - Clear the JSP engine's cache or work directory to ensure that the JSP engine picks up your changes. For example, to force Information Console's embedded servlet engine to use the changed files, delete all files from `C:\Program Files\Actuate\iPortal2\work` and clear the web browser's cache.

- Restart the JSP engine.
- If you do not see changes you made in a cascading style sheet file or a JavaScript file, clear the web browser's cache, then refresh the page.

Your changes appear in the web browser.

Locating existing pages and linking in new pages

To locate an existing page, navigate to that page and examine the URI in the address field of your browser. If the URI contains a JSP name, go to that JSP file. If the URI contains an action path, search `struts-config.xml` for that action path without the `.do` extension, or look up the action path in Chapter 4, "Actuate Information Console URIs."

An action path is a uniform resource identifier (URI) called directly by Information Console or by a user to access the Information Console functionality. `<context root>\WEB-INF\struts-config.xml` contains the action path specifications.

An action path specifies a JSP to use in response to user controls. An action path uses the results of an Action class to determine the next action path to use or the next JSP to display. Typically, an action class indicates one action path or JSP if the execution succeeds and a different action path or JSP if the execution causes an error. In the following code sample, if the `AcGetFolderItemsAction` JavaBean executes successfully, the next JSP to display is `<context root>\iportal\activePortal\private\filefolders\filefolderlist.jsp`:

```
<!-- Process getfolderitems -->
<action
  attribute="fileListActionForm"
  name="fileListActionForm"
  path="/getfolderitems"
  scope="request"
  type="com.actuate.activeportal.actions.AcGetFolderItemsAction"
  validate="false">
  <forward name="success"
    path="/iportal/activePortal/private/filefolders
      /filefolderlist.jsp" />
  <forward name="dashboard" path="/dashboard" redirect="true" />
</action>
```

In the preceding example, the path for an error result is not listed. This means that it defaults to the definition in the global forwards section of `struts-config.xml` as a when an error occurs:

```
<forward name="error" path="/iportal/activePortal/private/common
  /errors/errorpage.jsp"/>
```

To add a forward command that activates when the JavaBean returns another result, such as `viewbirt`, you can include a forward for that result to direct it accordingly, as shown in the following example:

```
<forward name="viewbirt"
  path="/iportal/activePortal/viewer/viewframeset.jsp"
  redirect="true" />
```

To add a new web page to Information Console, you change the navigation in `struts-config.xml` to use the new JSP or path. You can change an existing input page or forward page specification in an action path to your new page, or you can create a new action path that forwards to your page. If you create a new action path, you can change another action path to forward to your new path or you can modify or create links on web pages to specify your new action path. The following action path always navigates to `welcome.jsp` when another action path, link, or URL invokes it:

```
<!-- Process welcome -->
<action path="/welcome"
  forward="/iportal/activePortal/private/welcome.jsp"
  name="welcome">
</action>\
```

For more information on action paths and Jakarta Struts, go to the following URL:

<http://jakarta.apache.org/struts>

Obtaining information about the user and the session

Typically, new Actuate Information Console web pages need access to session information. Your application server and Information Console store information about the session that you can use in your web pages. Obtain the `serverURL`, `volume`, and other information from your application server using the JSP request variable, as shown in the following example.

```
String volume = request.getParameter("volume");
String serverURL = request.getParameter("serverurl");
String userId = request.getParameter("userid");
String thisReport = request.getParameter("report");
```

You can also obtain the context root path from your application server, as shown in the following code:

```
String contextRoot = request.getContextPath();
```

Additionally, Actuate Information Console stores a wide variety of information about the session in `UserInfoBean`. To access `UserInfoBean`, place the following line of code near the top of your JSP:

```
<jsp:useBean id="userinfobean"
  class="com.actuate.activeportal.beans.UserInfoBean"
  scope="session"/>
```

After this line, you can access information in the JavaBean by the appropriate get method. The most important method for new pages is the `getIportalid()` method. This method retrieves the user's authentication ID with the server. This ID is based on the server, volume, and user name supplied on the login page.

To write generic code, you need to determine whether your application is running. Information Console includes a utility class, `iPortalRepository`, that provides this information. To access this class in your JSP, place the following code at the head of your JSP:

```
<%@ page
    import="com.actuate.iportal.session.iPortalRepository" %>
```

Then use code similar to the following line to check the repository type:

```
boolean isEnterprise =
    iPortalRepository.REPOSITORY_ENCYCLOPEDIA.equalsIgnoreCase(
        userinfobean.getRepositoryType());
```

Use the authentication ID and the repository type to access the server with JSP custom Actuate tags and calls to Information Console beans, as shown in the following examples:

```
String authenticationID = userinfobean.getIportalid();
String folderPath = userinfobean.getCurrentfolder();
jobDetailURL += StaticFuncs.encode(userinfobean.getUserid());
com.actuate.reportcast.utils.AcLocale acLocale =
    userinfobean.getAcLocale();
TimeZone timeZone = userinfobean.getTimezone();
boolean isEnterprise =
    iPortalRepository.REPOSITORY_ENCYCLOPEDIA.equalsIgnoreCase(
        userinfobean.getRepositoryType());
String serverURL =
    ( isEnterprise | userinfobean.getServerurl() | " " );
String userVolume =
    ( isEnterprise | userinfobean.getVolume() | " " );
```

Customizing accessible files and page structure using templates

Actuate Information Console uses Jakarta Struts templates to simplify JSP code and customization. Information Console templates handle overall page organization, access to Jakarta Struts custom tag libraries, and access to common CSS and JavaScript files. The login page and landing page do not use a template. Table 2-4 describes the Information Console templates.

Table 2-4 Actuate Information Console Struts templates

Template	Method
dashboardtemplate.jsp	Used for BIRT 360 dashboards

Table 2-4 Actuate Information Console Struts templates

Template	Method
simpletemplate.jsp	Used for errors, confirmations, and other simple pages
template.jsp	Used by all other pages except the login and landing pages

Each Actuate Information Console skin has its own version of these templates, besides the dashboard template, in <context root>\iportal\activePortal\private\skins\<skin name>\templates. The set of templates in <context root>\iportal\activePortal\templates sets up several JavaBeans and then accesses the template of the same name for the user’s selected skin. The dashboard template is located in <context root>\dashboard\jsp, along with the dashboard JSP files.

Specifying a template and template elements

To use a template and template elements, a page uses the Jakarta Struts custom template tags, described in Table 2-5.

Table 2-5 Struts template tags

Template tag	Method
template:insert	Specifies the template to use
template:put	Specifies the text or file to use for a template element such as the name, banner, side menu, or content elements

The custom template tags define the JSPs to use for the template and the custom elements that the template specifies to build the user interface. For example, the template:insert tag in the following code applies querytemplate.jsp settings to the page. The first template:put tag accesses the localized string for the title of the page. The remaining template:put tags specify that the template use banner and content elements using the files specified in each tag.

The following tables show JSPs affected by template changes. Table 2-6 lists the Information Console templates and the pages that use them.

Table 2-6 Templates for JSPs

Template	JSPs in iportal\activePortal\private
simpletemplate.jsp	common\errors\errorpage.jsp customization\fileupload.jsp newrequest\newrequest2.jsp

(continues)

Table 2-6 Templates for JSPs (continued)

Template	JSPs in <code>iportal\activePortal\private</code>
<code>template.jsp</code>	<code>channels\channellist.jsp</code> <code>channels\channelnoticelist.jsp</code> <code>channels\channeloperationstatus.jsp</code> <code>channels\channelsubscribe.jsp</code> <code>customization\skinedit.jsp</code> <code>customization\skinmanager.jsp</code> <code>filesfolders\createfolder.jsp</code> <code>filesfolders\deletefilestatus.jsp</code> <code>filesfolders\filedetail.jsp</code> <code>filesfolders\filefolderlist.jsp</code> <code>filesfolders\privilege.jsp</code> <code>filesfolders\search\filefolderlist.jsp</code> <code>jobs\getjobdetails.jsp</code> <code>jobs\joboperationstatus.jsp</code> <code>jobs\selectjobs.jsp</code> <code>newrequest\submitjobstatus.jsp</code> <code>options\options.jsp</code>

About the dashboard template

The JSPs in `<context root>\dashboard\jsp` define the dashboard interface for the Actuate BIRT 360 option. The dashboard template applies to the banner and content when the dashboard is visible as defined by `dashboard.jsp`, as follows:

```
<template:insert template="/dashboard/jsp/dashboardtemplate.jsp">
  <template:put name="title" direct="true">
    <bean:message bundle="iportalResources" key="MSGT_BROWSER"
      arg0="<%= pageTitle %>" />
  </template:put>
  <template:put name="banner" content="<%= bannerUrl %>" />
  <template:put name="content" content="/dashboard/jsp/
    dashboardcontent.jsp" />
</template:insert>
```

The contents of a separate page or pages displayed on a dashboard use templates defined by that page's code, but the surrounding interface elements adhere to the dashboard template.

Changing a template

Make changes to all pages that use a particular template by changing only the template. You can add or remove lines in the template that make cascading style sheets, JavaScript files, and other resources accessible to all pages that use the template. Customize the overall structure of all pages that use a template by

moving, resizing, or removing the HTML, JSP, and Jakarta Struts code describing the layout of the web pages that use the template.

For example, the innerTable of <context root>\iportal\activePortal\private\skins\treeview\templates\template.jsp specifies various HTML commands and embedded Jakarta Struts tags that populate the content frame. The inner banner with the breadcrumb is in the top row. The second row contains the content page.

```
<table class="innerTable" border="0" cellspacing="0"
      cellpadding="0">
  <% if (!"false".equalsIgnoreCase(showBreadCrumb)) { %>
  <tr>
    <td class="allBreadcrumbs">
      <jsp:include page="<%= breadcrumb %>" flush="true" >
      <jsp:param name="fromDashboard" value="<%= fromDashboard %>" />
      <jsp:param name="showBanner" value="<%= showBanner %>" />
      <jsp:param name="showSideBar" value="<%= showSideBar %>" />
      <jsp:param name="showBreadCrumb" value="<%= showBreadCrumb %>"
      />
      </jsp:include>
    </td>
  </tr>
  <% } %>
  <tr>
    <td class="fileFolderListContent" valign="top">
      <div>
        <template:get name="content" flush="true"/>
      </div>
    </td>
  </tr>
</table>
```

The breadcrumb, or navigation trail, is a link or set of links. On a document page, the breadcrumb displays the repository and any folders and pages you access. Use any of these items as a link to return to that level. For a jobs or channels page, the breadcrumb supports direct access to a document page.

To implement the expandable tree, a frameset in <context root>\iportal\activePortal\private\skins\treeview\templates\template.jsp specifies the sidebar and content frames using HTML and embedded Jakarta Struts tags that define the content.

```
<FRAMESET cols="20%,80%" border="1"
  onload="if (typeof(bodyOnload) != 'undefined') bodyOnload();">
  <FRAME src="<html:rewrite page="<%= sidebar %>" />"
    name="<%=htmlSideFrameName%>"
    id="<%=htmlSideFrameId%>"
```

```

        scrolling="auto"
    />
    <FRAME src="<html:rewrite href="<%= contentURL %>" />"
        name="main" scrolling="auto"
    />
</FRAMESET>

```

Modifying existing content or creating new content

You can modify the content of an existing page or create new pages for linking in to your custom web application. Typically, a web page has one JSP that implements a template and another JSP to implements the content to display according to the template's structure. For example, the following code specifies that the template's content element on a web page uses the JSP code in <context root>\portal\activePortal\private\newrequest\newrequestpage.jsp:

```

<template:put name="content" content="/portal/activePortal
/private/newrequest/newrequestpage.jsp" />

```

The content JSP contains the code that creates the page-specific content and functionality. The newrequestpag.jsp contains code that places page-specific text, graphics, links, and other functionality on the page. You can use HTML code, JSP code, JSP built-in tags, Jakarta Struts tags, Actuate servlets, Actuate custom tags, Actuate JavaBeans, CSS, and JavaScript methods to obtain data and present information on the page. For information about how to use these features, see “Customizing an Information Console web application,” later in this chapter.

The default Actuate Information Console pages use HTML tables to provide formatting for each page. The tables are often nested. Individual files include other files that define elements, such as the <TABLE> declaration. As you modify the pages to suit your needs, verify that the Actuate Information Console pages for tasks, such as logging in, listing folders and files, and viewing and requesting reports appear correctly in your web browser.

When using relative hyperlinks in your HTML code, ensure that any files to which you refer are available to Actuate Information Console. Information Console resolves relative hyperlinks from the context root. For example, in the standard Information Console installation, the following code refers to an images directory at the same level as the Information Console context root directory:

```

<A HREF=" ../images/myimage.gif">

```

All Actuate Information Console requests require action paths to have certain names. Similarly, the action paths require JSP files to have certain names and to reside in a particular directory under the context root. Do not rename the default files provided with Information Console without making the corresponding change to struts-config.xml. If you do not change the file name consistently in all places, Information Console cannot locate your custom files.

Modifying global style elements

Although JSPs can use HTML to set colors, fonts, and other stylistic elements directly, the JSPs also use cascading style sheets (CSS), templates, and shared images to control the global styles of an Information Console web application. To modify the appearance of the entire Information Console web application, change global style elements.

Global styles can change more than the appearance of Actuate Information Console. For example, to view search results with HKSCS characters in an English locale, change the `.searchresultlink` style's font from Arial to MingLiU_HKSCS. This style change only affects the search results.

Customizing Actuate Information Console using skins

Actuate Information Console skins support customizing the Actuate Information Console colors, fonts, and images in the graphical user interface (GUI) for the pages in My Documents, BIRT Studio, and the interactive viewer for BIRT reports. A skin consists of images, cascading style sheets, JavaScript, and template files used to define the GUI. Actuate Information Console installs with three skins. Only users with the Administrator functionality level can customize skins.

Using skins

To select a different Information Console skin, a user chooses Options, then selects a name from the Skin drop-down list, as shown in Figure 2-8.

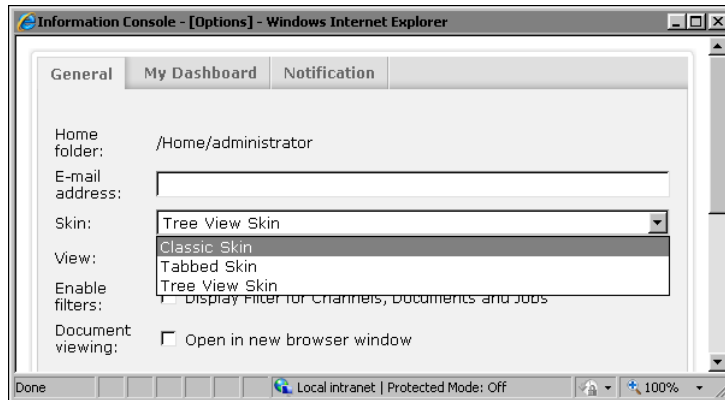


Figure 2-8 Default skins choices

Actuate Information Console provides three default skins:

- Use the Classic skin to view Documents, My Jobs, and Channels as buttons in the side menu, as shown in Figure 2-9.

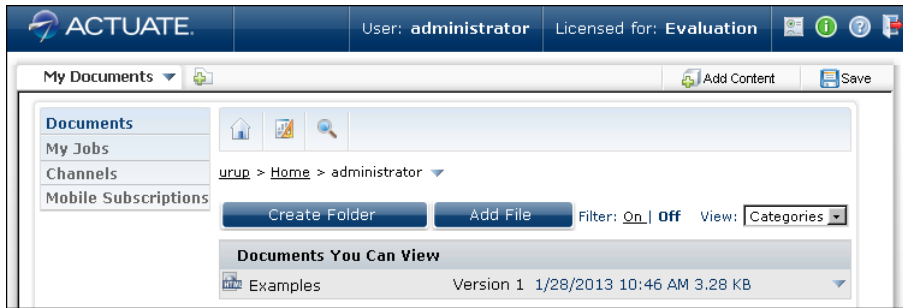


Figure 2-9 Classic skin

- Use the Tabbed skin to view Documents, My Jobs, and Channels as tabs on the banner at the top of the page, as shown in Figure 2-10.

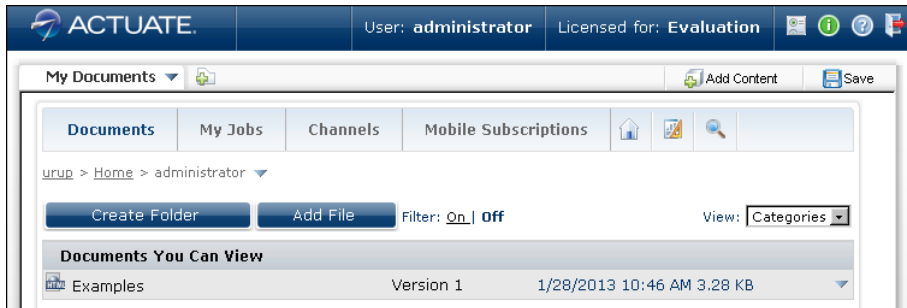


Figure 2-10 Tabbed skin

- Use the Treeview skin to view Documents, My Jobs, and Channels in the side menu as a hierarchical view. The folders view starts from the root folder of an Encyclopedia volume. This hierarchical view is similar to that of Windows Explorer, as shown in Figure 2-11, and is the default skin. The Treeview skin does not support placement in an iFrame.

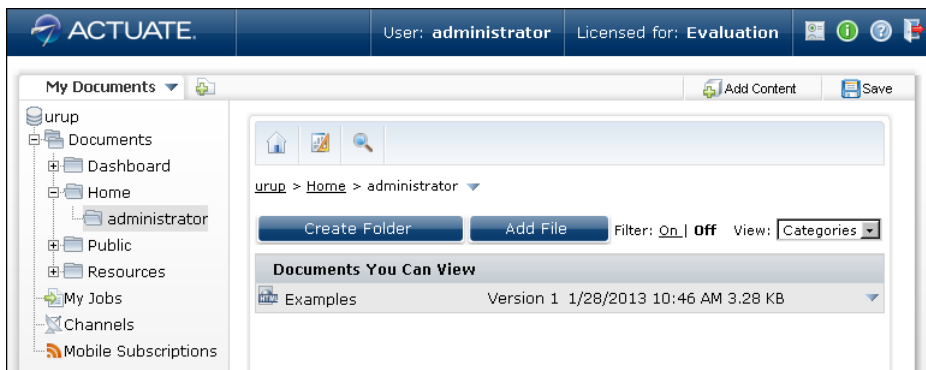


Figure 2-11 Treeview skin

Managing skins using the skin manager

Users with the Administrator functionality level manage skins for all users. The skin manager controls skins and their settings. To access the skin manager, choose Customization on the Information Console banner as shown in Figure 2-12.

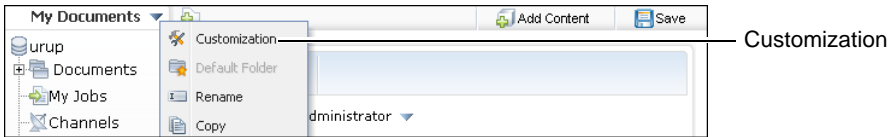


Figure 2-12 Information Console banner showing Customization menu option

The default skin manager looks like the one in Figure 2-13.

Skins > Manager			
Name	Description	Default	Public
classic	Classic Skin		<input checked="" type="checkbox"/>
tabbed	Tabbed Skin		<input checked="" type="checkbox"/>
treeview	Tree View Skin		<input checked="" type="checkbox"/>

Figure 2-13 Skin manager showing the default skins

Table 2-7 describes the features of the skin manager.

Table 2-7 Skin manager functionality

Feature	Description
Clone	Adds a copy of the skin to the table as private.
Customize	Displays the Skin—Customize page to allow customizing for that skin. The skins shipped by default with Actuate Information Console cannot be customized.
Default	Selects the skin used for new users by default without affecting existing users. Setting a skin to Default makes it public and disables its Public check box.
Delete	Deletes the skin after confirmation. Skins shipped with Actuate Information Console and the default skin cannot be deleted. To delete the current default skin, first choose another skin as the default.
Preview	Applies the skin immediately. When the current session times out, the skin reverts back to the user's original skin. The user's current skin is shown in bold text.
Public	Makes the skin available for all users by adding the skin to the list on the Options page. If a public skin becomes private, users using the skin revert to the default skin. The default skin is always public.

Customizing and cloning skins

Actuate Information Console ships with three standard skins. You cannot customize the standard skins. You can customize any skin clone, or copy, you create. The skins that Information Console provides may be modified during an upgrade, but any skin you create is preserved during upgrades. To customize any of the three standard skins, clone the skin to create a copy, and then customize the clone. When cloning a skin, select the skin that is closest to the required appearance. Perform additional customizations to a cloned skin at any time.

Table 2-8 lists the GUI components of cloned skins that you can customize.

Table 2-8 Customizable components of skins

Item	Customizable components
Colors	Banner, footer, side menu, tabbed dialogs, pop-up menus, viewer, templates
Fonts	Multiple font families in order of preference
General	Skin description that appears on the Options page
Images	Banner logo, My Folder icon, volume icon, open and closed folder icons

The skin description appears on the Options page to identify the skin to users.

Colors are grouped into categories according to the GUI area they affect. Table 2-9 lists ways to specify colors.

Table 2-9 Techniques for specifying colors

Specification	Description
Color code	Type a standard HTML color or hexadecimal RGB value.
Red Green Blue	Type individual RGB values, from 0 to 255.
Pick a color	Select from a palette of available colors.

To customize images for a skin, upload GIF or JPEG files to Actuate Information Console to replace the existing images. Images are grouped in categories by their GUI component. The categories and the images that you can replace depend upon the type of skin. For example, more images are available to customize in a skin based on the Treeview than a skin based on the Classic skin. Icon images must be consistent in size with the images they replace. Most icons supplied with Actuate Information Console are either 32x32 or 16x16 pixels.

After making changes to a skin, use the preview functionality to view different Actuate Information Console pages to show the skin's current appearance. By checking multiple pages, you identify the areas that need modification.

How to clone a skin

Use the following procedure to create a new skin, based on an existing skin.

- 1 Log in to the documents web pages as an administrator-level user. Choose Customization.
- 2 In the skin manager, choose Clone on an existing skin, as indicated in Figure 2-14.



The screenshot shows a web interface titled "Skins > Manager". It contains a table with four columns: "Name", "Description", "Default", and "Public". There are three rows of skins: "classic", "tabbed", and "treeview". Each row has four links: "Customize", "Clone", "Delete", and "Preview". An arrow points from the word "Clone" in the text to the "Clone" link in the "treeview" row.

Name	Description	Default	Public	
classic	Classic Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview
tabbed	Tabbed Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview
treeview	Tree View Skin	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview

Figure 2-14 Clone functionality for a skin

- 3 At the prompt, type a name for the new skin. Choose OK. The new skin appears in the list of available skins, as shown in Figure 2-15. Do not select Public or Default until you have finished the skin development.



The screenshot shows the "Skins > Manager" interface with the same table as Figure 2-14, but with an additional row at the bottom: "Clone of treeview". This new row has "Clone of treeview" in the Description column, the "Default" radio button is unselected, "Public" is unchecked, and it has the same four links: "Customize", "Clone", "Delete", and "Preview".

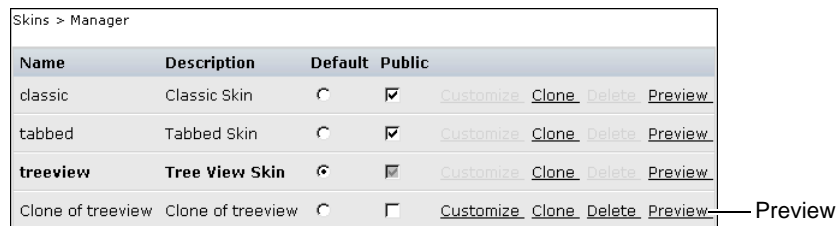
Name	Description	Default	Public	
classic	Classic Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview
tabbed	Tabbed Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview
treeview	Tree View Skin	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview
Clone of treeview	Clone of treeview	<input type="radio"/>	<input type="checkbox"/>	Customize Clone Delete Preview

Figure 2-15 The skin manager showing a cloned skin

How to customize a skin

The following procedure customizes the skin created in “How to clone a skin,” earlier in this chapter.

- 1 In the skin manager, on the skin to change, choose Preview, as indicated in Figure 2-16. The appearance of Actuate Information Console pages changes to match the selected skin.



The screenshot shows the "Skins > Manager" interface with the same table as Figure 2-15, but with an additional row at the bottom: "Clone of treeview". This new row has "Clone of treeview" in the Description column, the "Default" radio button is unselected, "Public" is unchecked, and it has the same four links: "Customize", "Clone", "Delete", and "Preview". An arrow points from the word "Preview" in the text to the "Preview" link in the "Clone of treeview" row.

Name	Description	Default	Public	
classic	Classic Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview
tabbed	Tabbed Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview
treeview	Tree View Skin	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	Customize Clone Delete Preview
Clone of treeview	Clone of treeview	<input type="radio"/>	<input type="checkbox"/>	Customize Clone Delete Preview

Figure 2-16 Preview functionality for a skin

- 2 On the skin to change, choose Customize, as indicated in Figure 2-17.

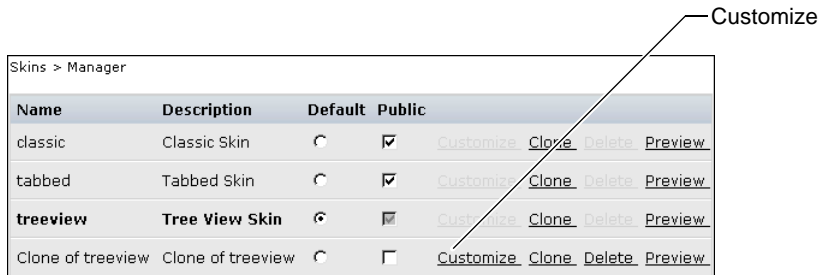


Figure 2-17 Customize functionality for a cloned skin

- 3 On Skins—Customize—General, change the skin description to a unique value that conveys meaning to your users, as shown in Figure 2-18.

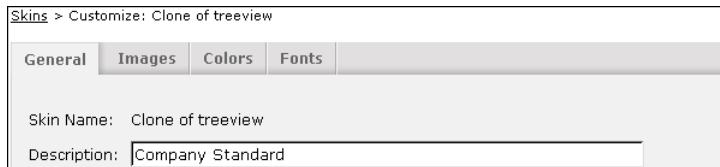


Figure 2-18 General pane for skin customization

- 4 Select Images. The Images pane appears, as shown in Figure 2-19. Choose a category name to see the images in that category.

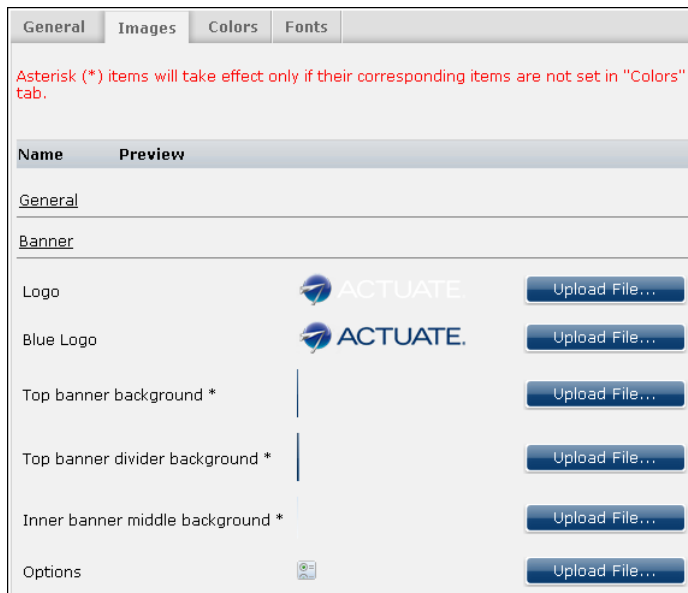


Figure 2-19 Images pane for skin customization

- 5 Select Colors. The Colors pane appears, as shown in Figure 2-20. The categories shown depend upon the type of skin. Choose a category name to toggle between showing and hiding the list of colors in that category.

Name	Color Code	Red	Green	Blue
<u>General</u>				
Landing and Login pages background *		0	0	0
Body background	White	255	255	255
Text Color	black	0	0	0
Panel background color	#DADADA	218	218	218
Listing content background color	#E8E8E8	232	232	232
Listing content highlight color	white	255	255	255
Panel border color	#b5b29c	181	178	156
Breadcrumbs background color	white	255	255	255

Figure 2-20 Colors pane for skin customization

- 6 Select Fonts. The Fonts pane appears, as shown in Figure 2-21. On Name, select General. Font Family appears. Specify one or more font families to use. Actuate Information Console uses the first font in the list found on the machine where Actuate Information Console is deployed.

Name	Preview	Font family in order of preference
<u>General</u>		
Font Family	Sample Text	Verdana, Arial, Helvetica, sans-serif

Figure 2-21 Fonts pane for skin customization

Choose OK.

- 7 To make the new skin available to all users, on the Skins—Manager page, select Public for your new skin.
- 8 To make the skin the default skin for all users, select Default. Figure 2-22 shows a custom skin, Clone of classic, based on the Classic skin.

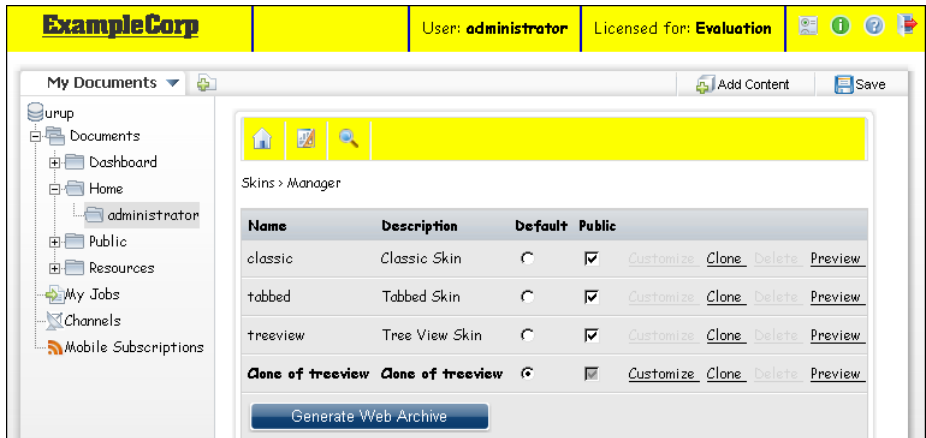


Figure 2-22 An example of a custom skin

Understanding style definition files

Additional style definitions for each provided skin come from `<context root>\iportal\activePortal\private\skins\<skin name>\css\skinstyles.css`. Add more styles to this file if you want the style definitions to take effect for only a particular skin. Information Console's JSPs typically link these styles in the following order:

- `<context root>\css\allstyles.css`

```
<LINK href="<html:rewrite page="/css/allstyles.css"/>"
      type="text/css" rel="stylesheet">
```
- `<context root>\iportal\activePortal\private\skins\<skin name>\css\skinstyles.css`

```
<LINK
  href="<ap:skinResource resource="/css/skinstyles.css" />"
  type="text/css" rel="stylesheet" >
```
- Style specifications from the customization web pages


```
<STYLE>
  <bean:write
    name="userinfobean" property="skinConfig.cssCode" />
</STYLE>
```

If a style is defined in more than one of these files, the JSP engine uses the definition in the last file that contains the style. Thus the settings you specify in the customization web pages override any other CSS files.

`allstyles.css` contains additional style definitions for the Actuate Information Console application. Modify `allstyles.css` to change any style definitions that are not handled within the customization web pages or the `<context root>\iportal`

\activePortal\private\skins\<skinname>\css\skinstyles.css file. Changes to a style in allstyles.css affects all Information Console skins except the parameters page unless the customization web pages or a skin's skinstyles.css file override it. To customize the parameter component, modify the style definitions in the <context root>\css\parameter.css file.

How to test and modify styles depending on the browser type

- 1 Near the top of your JSP, link in the allstyles.css style sheet:

```
<LINK href="<html:rewrite page="/css/allstyles.css"/>"
      type="text/css" rel="stylesheet" >
```

- 2 After this line, link in the style sheet located in the current skin's css directory:

```
<LINK href="<ap:skinResource resource="/css/skinstyles.css" />"
      type="text/css" rel="stylesheet" >
```

- 3 Use the Jakarta Struts bean:write custom tag to generate and include style definitions for styles defined using the skin customization pages:

```
<STYLE>
  <bean:write
    name="userinfobean" property="skinConfig.cssCode" />
</STYLE>
```

- 4 If the skin customization styles contain any settings that do not work in a specific browser, you can override them individually.

Specifying colors and fonts

Specify fonts and colors for styles in the customization web pages or in the cascading style sheets. Specify colors using the following methods:

- Using a color name such as navy, yellow, or teal, as shown in the following example:

```
color: Yellow;
```

- Using hexadecimal notation to set the amount of red, green, and blue to use in the color.

```
#FFFF00
```

- Using decimal notation to set the amount of red, green, and blue to use in the color. In the customization web pages, fill in the value for red, green, and blue in the corresponding fields. In a CSS file, use a call to the rgb() method, as shown in the following example:

```
color: rgb(156, 207, 255);
```

How to change the font style of a single item

To change Actuate Information Console pages to display the user, system name, and volume in 12-point italic Comic Sans MS font:

1 In a text editor, open <context root>\css\allstyles.css.

2 Locate the following string:

```
bannerTextArea
```

There are two instances of the string bannerTextArea. The first is part of the definition for all the banner styles. This definition sets the banner styles' common attributes. The second instance sets the attributes for bannerTextArea only and looks like the following text:

```
.bannerTextArea {  
    color: white;  
    font-size: 10pt;  
    text-align: left;  
    white-space: nowrap;  
}
```

3 Modify the code that follows the bannerTextArea definition to change the font as shown in the following code:

```
.bannerTextArea {  
    color: white;  
    font-family: Comic Sans MS;  
    font-size: 11pt;  
    font-style: italic;  
    text-align: left;  
    white-space: nowrap;  
}
```

4 Save and close the CSS file.

5 Refresh your web browser to view the changes. Figure 2-23 shows the new appearance of the banner.



Figure 2-23 Appearance of customized Information Console banner

Customizing page styles for BIRT Studio

To customize BIRT Studio pages, use the files in <context root>\iportal\bizRD\styles. This directory includes the following customizable CSS files:

- accordion.css defines styles for the report design area of the page, which displays the Available Data, Report Template Items, and other selectable tree views.
- dialog.css defines styles for dialog boxes that have shared characteristics, including the dialog boxes for template selection, file browsing, calculations, parameters, and so on.

- dialogbase.css defines the style of dialog containers, such as the button style, the Close icon style, and so on.
- title.css defines styles for the title bar of BIRT Studio pages.
- toolbar.css defines styles for the toolbar.
- wrcontextmenu.css defines the styles for BIRT Studio context menus.

Another file in this directory, webreporting.css, is not customizable.

For more information about using cascading style sheets, access the following URL:

<http://www.w3.org/Style/CSS/>

Modifying graphic images

Information Console pages use images for the company logo in the banners, on the side menu, and for the background. Some pages use additional images that are related to their content. You can also add new images on pages.

Certain images are most easily changed by customizing a skin. You can customize the company logo and the My Folder icon for all skins. In addition, you can customize the open and closed folder icons and volume icon for a skin that is cloned from the Treeview skin. These and all other images that you can customize reside in <context root>\portal\activePortal\private\skins\<skin name>\images. Update these images by using the skin customization pages to use new graphic files instead of changing the supplied graphic files. Customizing the images described in Table 2-10 affects most Information Console web pages.

Table 2-10 Images in Information Console skins

Skins	Default image file	Description
All	logo.gif	The company logo to use in the banners
All	homefoldericon.gif	The image to use beside the My Folder link
Treeview	closedfoldericon.gif	The image to use to indicate a unexpanded folder in the hierarchical view of the volume and folders
Treeview	foldericon.gif	The image to use to indicate an expanded folder in the hierarchical view of the volume and folders
Treeview	volume_icon.gif	The image to use to indicate a volume in the hierarchical view of the volume and folders

An additional image of interest is <context root>\portal\activePortal\private\skins\<skin name>\images\background.gif. The Classic skin and its clones use this image to provide the background for every page. This image is one pixel high

and 1280 pixels long, and is copied as necessary to fill the page. Change the contents of this image file to modify the background of a Classic skin clone.

All other images reside in <context root>\iportal\activePortal\images. This set of images provides the features on the side menu in the Classic skin and the tree in the Treeview skin. Update these feature images by changing the corresponding feature definition in the \iportal\WEB-INF\functionality-level.config file.

Other images are referenced by hard-coded path and file names in JSP and JavaScript files, such as the icons in <context root>\iportal\activePortal\private\filesfolders\views\categories.jsp. For example, categories.jsp specifies the location and filename, <context root>\iportal\activePortal\images\detailicon.gif, a magnifying glass icon that is used to obtain more details about a document or other item in a list. When you change the location or replace an image with a new file, you must update the JavaScript and JSP files that use them. Alternatively, make a backup copy of the original image and then reuse the original name for your new image. By reusing the original name, you do not need to make any changes in the JSP and JavaScript files using the image.

Part Two

Actuate Information Console reference

Actuate Information Console configuration

This chapter contains the following topics:

- About Information Console configuration
- Configuring the Information Console web application
- Configuring the connection to iHub
- Configuring the BIRT Viewer and Interactive Viewer
- Configuring BIRT Studio
- Configuring BIRT Data Analyzer

About Information Console configuration

The Information Console application is configured using files in the context root's WEB-INF directory. For example, the web.xml configuration file for your context root is located:

```
<context root>\WEB-INF\web.xml
```

Table 3-1 lists the configuration files discussed in this chapter.

Table 3-1 Information Console configuration files

File	Features	Description
erni_config.xml	BIRT Studio	Configures BIRT Studio functionality
functionality-level.config	Information Console	Configures the Information Console user interface by iHub security roles
iv_config.xml	BIRT Viewer	Configures BIRT Viewer user interface
localemap.xml	All	Configures languages and locales
TimeZones.xml	All	Configures time zones
volumeProfile.xml	All	Consolidates iHub volume connection information into a single handle, hiding iHub volume details in a URL
web.xml	All	Configures features of Information Console including security, networking, caching, labeling and storage

Configuring the Information Console web application

Information Console provides the ability to organize, run, and view reports. You configure the user interface, logging, and caching for Information Console using web.xml.

Configuring Information Console using web.xml

Web.xml contains parameters that control Information Console features. Table 3-2 describes the configuration parameters for the Information Console application.

Table 3-2 Actuate Information Console web.xml parameters

Parameter name	Description
BIRT360PLUS_URL	URL for the BIRT 360 plus web resources required for performance analytics gadgets, available in the gadget gallery.
BIRT_RENDER_FORMAT_EMITTER_ID_MAPPING	<p>Specifies which emitter will be used for a specific BIRT report. Valid entries are of the format "render_format:emitter_ID" separated by a semicolon. The default value is:</p> <p>html:org.eclipse.birt.report.engine.emitter.html;xhtml:com.actuate.birt.report.engine.emitter.xhtml;pdf:org.eclipse.birt.report.engine.emitter.pdf;postscript:org.eclipse.birt.report.engine.emitter.postscript;xls:com.actuate.birt.report.engine.emitter.xls;ppt:org.eclipse.birt.report.engine.emitter.ppt;pptx:com.actuate.birt.report.engine.emitter.pptx;doc:org.eclipse.birt.report.engine.emitter.word;docx:com.actuate.birt.report.engine.emitter.docx</p>
CACHE_CONTROL	<p>Specifies how a web browser caches information using one of the following values:</p> <ul style="list-style-type: none">■ NO-CACHE indicates that the browser does not cache information and forwards all requests to the server. With NO-CACHE, the back and forward buttons in a browser do not always produce expected results, because choosing these buttons always reloads the page from the server. If multiple users access Information Console from the same machine, they can view the same cached data. Setting CACHE_CONTROL to NO-CACHE prevents different users viewing data cached by the browser.■ NO-STORE indicates that information is cached but not archived.■ PRIVATE indicates that the information is for a single user and that only a private cache can cache this information. A proxy server does not cache a page with this setting.■ PUBLIC indicates that information may be cached, even if it would normally be non-cacheable or cacheable only within an unshared cache.■ Unset (no value) is the default value. The browser uses its own default setting when there is no CACHE_CONTROL value. <p>Caching information reduces the number of server requests that the browser must make and the frequency of expired page messages. Caching increases security risks because of the availability of information in the cache. For additional information about cache control, see the HTTP/1.1 specifications.</p>

(continues)

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
CONNECTION_TIMEOUT	Controls how many seconds Actuate Information Console waits for a request to complete before dropping the connection to the application server or Actuate BIRT iHub. Set this value to limit wait times. The default value is 0, meaning the connection is never dropped.
COOKIE_DOMAIN	Specifies the host name of the server setting the cookie. The cookie is only sent to hosts in the specified domain of that host. The value must be the same domain the client accesses. Information Console automatically sets this parameter. For example, if the client accesses http://www.actuate.com/iportal/login.do , the domain name is <code>actuate.com</code> .
COOKIE_ENABLED	Indicates whether to use cookies to store information between user logins. The default value is <code>True</code> . If <code>False</code> , Information Console does not use cookies. Without cookies, many Information Console features are unavailable or do not persist across sessions. For example, without cookies, user name, language, and time zone settings always use their default values when a new browser session begins.
COOKIE_SECURE	Indicates whether to access and write cookies securely. If <code>true</code> , cookies are only written if a secure connection, such as <code>HTTPS</code> , is established. The default value is <code>false</code> , which enables cookies for all connection types.
DEFAULT_LOCALE	Specifies the default locale. Information Console sets this parameter value during installation. The locale map is <code><context root>\WEB-INF\localemap.xml</code> .
DEFAULT_PAGE_BREAK_INTERVAL	Specifies the number of rows to display in one page when viewing a report. If set to 0, there are no page breaks.
DEFAULT_TIMEZONE	Specifies the default time zone. Information Console sets this parameter value during installation. The time zone map is <code><context root>\WEB-INF\TimeZones.xml</code> .
MOBILE_APP_DOWNLOAD	The URL target of the download button displayed by Information Console when viewed in mobile/touch device.
ENABLE_CLIENT_SIDE_REDIRECT	Specifies whether URL redirection is done on the client side or the server side. Set the value to <code>True</code> for client side redirection. The default value is <code>False</code> . For more information about URL redirection, see “Using proxy servers with Actuate Information Console” in Chapter 1, “Introducing Actuate Information Console.”
ENABLE_DEBUG_LOGGING	Indicates whether to record debugging messages in a log file called <code>Debug.log</code> . Set the value to <code>True</code> to enable debug messages in the log file. The default value is <code>False</code> .

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
ENABLE_ERROR_LOGGING	Indicates whether to log errors. This parameter's default value is True, which enables error logging. If you set this parameter to True, Information Console creates two error log files: <ul style="list-style-type: none"> ■ Admin.log records general errors. ■ Soapfault.log records iHub communication errors.
ENABLE_JUL_LOG	Indicates whether to log Information Console activity. This parameter's default value is TRUE, which enables logging. If you set this parameter to TRUE, Information Console creates log files named reportService.<Service number>.<System name>.<Information Console start up time stamp>.<File number>.log.
ERROR_LOG_FILE_ROLLOVER	Specifies the time period to wait before starting a new log file. Options are Daily, Monthly, Weekly, and Yearly. The default value is Monthly.
EXECUTE_DASHBOARD_GADGET_GENERATION_WAIT_TIME	Specifies the time to wait, in seconds, for a gadget to generate when running a dashboard design file. This parameter's default value is 2 seconds.
EXECUTE_REPORT_WAIT_TIME	Specifies the time to wait, in seconds, for a report to execute. This parameter's default value is 20 seconds. For more information about the wait time parameter, see "execute report page" in Chapter 4, "Actuate Information Console URIs."
FILES_DEFAULT_VIEW	Specifies the default view for the files and folders list using one of the following values: <ul style="list-style-type: none"> ■ Categories, the default, displays files organized in rows by type. ■ Detail displays files organized in rows by name. ■ List displays files organized in columns with small icons. ■ Icon displays files organized in columns with large icons.
FORCED_GC_INTERVAL	Indicates the length in seconds of the interval that the Information Console application waits between forced garbage collections. To disable garbage collection, set this parameter to 0, the default value. Use this parameter to tune application server performance. 600 seconds is the recommended value. If the value is too low, the application server performs garbage collection too frequently, slowing the system. If the value is too high, you waste memory. If disabled, the application server controls garbage collection.
GADGET_GENERATION_WAITING_TIME	Specifies the time to wait, in seconds, for an individual gadget to generate. This parameter's default value is 10 seconds.

(continues)

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
IDAPI_TIMEOUT	Specifies the number of seconds to wait for a SOAP message response. This value must be larger than the maximum time necessary to run a report design or Information Console generates a time-out error for some reports. Its default value is 7200.
INSTALL_MODE	Indicates whether Information Console is installed with iHub. The value is set when Actuate Information Console is installed. Do not change this setting.
JUL_LOG_CONSOLE_LEVEL	The level of Information Console activity to log to the console. Valid values are OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, in order of the number of messages to log. The default value is OFF.
JUL_LOG_FILE_COUNT	Specifies the number of log files for a particular time stamp, if the value of ENABLE_JUL_LOG is TRUE.
JUL_LOG_FILE_LEVEL	The level of Information Console activity to log in a file. Valid values are OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, in order of the number of messages to log. The default value is WARNING.
JUL_LOG_FILE_SIZE_KB	The maximum size, in kilobytes, for an Information Console activity log file. When a log file reaches this size, Information Console creates a new log file and increments its file number. If the log file number reaches the value of JUL_LOG_FILE_COUNT, Information Console resets the file number to zero and overwrites the first log file for the time stamp.
LOG_FILE_LOCATION	Indicates which directory contains the log files. If the value is not an absolute directory path name, Actuate Information Console locates the directory in the Information Console home directory. The default value is logs in the Information Console home directory.
LOGIN_TIMEOUT	Specifies the number of seconds to wait before a session times out. The minimum login time-out is 300 seconds. The maximum value is equivalent to java.lang.Long. Its default value is 1200 seconds.
MAX_BACKUP_ERROR_LOGS	Specifies the maximum number of backup error log files to keep. The default value is 10.
MAX_LIST_SIZE	Limits the number of items returned when getting folder items, jobs, job notices, scheduled jobs, and channels to reduce network traffic. The default value is 150.
PRELOAD_ENGINE_LIST	List of engines to load when Information Console starts. Valid values are birt and ess. Default value is "birt, ess" which indicates both.

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
PROGRESSIVE_REFRESH	Controls the interval in seconds at which an Actuate report refreshes itself when running a progressive report. The report refreshes first after 15 seconds, then after 60 seconds, and then after the PROGRESSIVE_REFRESH interval. If the value is less than 60, Actuate Information Console uses 60 seconds. This parameter's default value is 1800 seconds.
PROGRESSIVE_VIEWING_ENABLED	Specifies whether a paginated report starts to display in the browser as soon as the first page has been generated. Valid values are true and false. The default value is true.
PROXY_BASEURL	Indicates a proxy server's URL if the network uses one between Information Console and the client. The default value is blank, which indicates that the network does not use a proxy server.
SECURITY_ADAPTER_CLASS	Specifies the fully qualified class of the security adapter, which must extend com.actuate.iportal.security.iPortalSecurityAdapter, that controls access to Actuate Information Console functionality. The default value is no name.
SESSION_DEFAULT_PARAMETER_VALUE_ID	Specifies the name of the object that stores the HTTP session-level report parameters. This object is an instance of the com.actuate.parameter.SessionLevelParameter class, which is extensible. The default value is SessionDefaultParameterValue.
sessionTimeout	The number of milliseconds the Information Console Ajax Proxy maintains an idle session. The default value is 5000.
TRANSIENT_STORE_MAX_SIZE_KB	Limits the amount of disk space that Actuate Information Console uses for temporary files. The default value is 102400, which is 100 MB.
TRANSIENT_STORE_PATH	Path to Actuate Information Console transient files. The default value is set when Information Console is installed. When deploying more than one context root or separate server, set a unique path for each.
TRANSIENT_STORE_TIMEOUT_MIN	Specifies, in minutes, how long to retain Actuate Information Console transient files. The default value is 40, which is 40 minutes.
UPLOAD_FILE_TYPE_LIST	Specifies the valid file types, by extension, for upload with Information Console. The default value is blank, which indicates any file type.
UPLOAD_SECURITY_MANAGER	Specifies the fully qualified class of the security adapter, which must extend com.actuate.iportal.security.IUploadSecurityAdapter, that controls access to the upload functionality. The default value is no name.

Configuring Information Console using volumeProfile.xml

Information Console uses a volume profile to access a specific iHub instance and Encyclopedia volume. Because the volume profile conceals the iHub and Encyclopedia volume values from the users of Information Console, the system administrator can change the location of these resources without affecting the URLs accessed by the users. To access iHub resources using a volume profile, add a `__vp=ProfileName` parameter to the URL.

Customize volume profiles by creating or modifying entries in the following file:

```
<context root>\WEB-INF\volumeProfile.xml
```

For example, the following is a volume profile definition for the server1 server:

```
<VolumeProfiles>
  <Profile>
    <Default>true</Default>
    <ProfileName>server1</ProfileName>
    <RepositoryType>enterprise</RepositoryType>
    <ServerUrl>http://server1:8000</ServerUrl>
    <Volume>volume1</Volume>
    <DashboardTemplatePath></DashboardTemplatePath>
  </Profile>
</VolumeProfiles>
```

- `<ProfileName>` is the name of this profile.
- `<RepositoryType>` has one of two values, either `enterprise` or `workgroup`.
- `<ServerUrl>` contains the iHub URL, for example, `http://server1:8000`. If `RepositoryType` is `workgroup`, `ServerUrl` is ignored.
- `<volume>` is the volume name. If `RepositoryType` is `workgroup`, volume is ignored.
- `<Default>` is optional. Valid values are `true` and `false`. A value of `true` sets this profile as the default volume profile, and the server and volume in this profile is used if no volume profile is provided in the URL. Information Console handles only the first profile with default set to `true` as the default profile.
- `<DashboardTemplatePath>` is optional. This repository path is the location of the dashboard file that loads when a user creates a new dashboards.

To make a new profile available to Information Console, add a new `<Profile>` element to the list in `<VolumeProfiles>` in `volumeProfile.xml`. Then, restart Information Console. For example, the following profile accesses the `volume2` volume on the `server2` server:


```

<Profile>
  <Default>false</Default>
  <ProfileName>server2</ProfileName>
  <RepositoryType>enterprise</RepositoryType>
  <ServerUrl>http://server2:8000</ServerUrl>
  <Volume>volume2</Volume>
  <DashboardTemplatePath></DashboardTemplatePath>
</Profile>

```

Using a volume profile defined in volumeProfile.xml

Information Console connects to the server and volume defined by the default volume profile entry when the URL does not include the `__vp` parameter or the volume parameter. For example, to connect to the default volume and server, use the following URL:

```
http://infoconsole:8900/portal/getfolderitems.do?userid=userName
&password=validPassword
```

Information Console connects to the server and volume defined by a volume profile when the URL contains a `__vp` parameter with a valid profile name and the URL does not have a volume parameter. For example, to connect to volume2 on server2 defined by the volume profile example above, use the following URL:

```
http://infoconsole:8900/portal/getfolderitems.do?userid=userName
&password=validPassword&__vp=server2
```

Overriding the volume specified in a volume profile

Information Console supports using a volume profile to access an Encyclopedia volume even if the volume is not specified in a volume profile definition. To override the volume specified by a volume profile, add the volume parameter to the URL. A URL with a valid `__vp` parameter and volume parameter connects to the server in the volume profile, but the volume assigned to the volume parameter. For example, to connect to volume3 on server2 explicitly, use the following URL:

```
http://infoconsole:8900/portal/getfolderitems.do?userid=userName
&password=validPassword&__vp=server2&volume=volume3
```

If the URL contains a volume parameter but not a `__vp` parameter, Information Console connects to the server in the default volume profile and the volume assigned to the volume parameter. For example, to connect to volume3 on the default server, use the following URL:

```
http://infoconsole:8900/getfolderitems.do?userid=userName
&password=validPassword&volume=volume3
```

Understanding temporary volume profiles

If a request URL contains `serverurl`, `repositorytype`, or `volume` parameters not defined in `volumeProfile.xml`, Information Console generates a temporary profile name for this set of volume properties. A temporary name is not persistent and is lost every time the application restarts. If the request URL does not contain `serverurl`, `volume`, and `repositorytype` parameters, Information Console uses the default profile for the request URL. If there is no default profile defined, Information Console generates a temporary server profile having a random name and uses `SERVER_DEFAULT`, `DEFAULT_VOLUME`, and `REPOSITORY_TYPE` defined in `WEB-INF/web.xml` as the default values for `serverurl`, `volume`, and `repositorytype`.

Configuring Information Console functionality levels with `functionality-level.config`

A functionality level defines which Information Console user interface features are visible and usable by members of an Encyclopedia volume security role or roles. By default, every user can access all functionality levels. Functionality levels corresponding to iHub security roles like Intermediate, Advanced, and Administrator, are provided in the comments of the `functionality-level.config` file. For example, by default every functionality level shows Log out, Options, and Help links on the Information Console banner.

The Intermediate and Advanced levels add a Search link to the documents page and the capability to add tabs, and the Administrator level adds a Customization link, as shown in Figure 3-1.

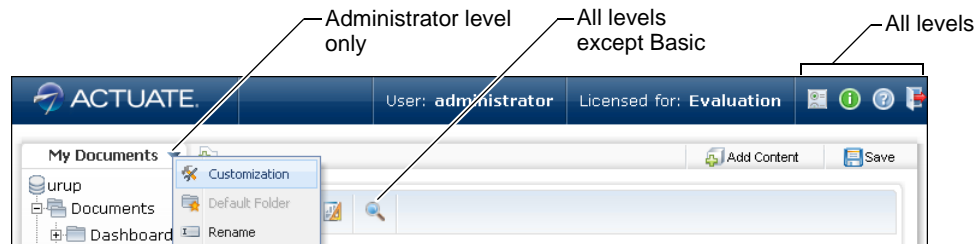


Figure 3-1 The banner appearance for a user at the Administrator functionality level

Actuate Information Console provides five functionality levels by default. Four functionality definitions specify a corresponding Encyclopedia volume security role that provides access to that functionality level. Table 3-3 shows the functionality levels and their corresponding security roles. The Administrator level is the Information Console Administrator, not the Encyclopedia volume administrator.

Table 3-3 Information Console default functionality levels and the corresponding Encyclopedia volume security roles

Functionality level	Security role
Basic	All (The All role includes all users.)
Intermediate	Active Portal Intermediate
Advanced	Active Portal Advanced
Administrator	Active Portal Administrator

Customize a functionality level by creating or modifying entries in the following file:

```
<context root>\WEB-INF\functionality-level.config
```

When modifying the configuration file, ensure that functionality levels in the configuration file specify a corresponding security role to enable access to that functionality level. You can modify the built-in levels but you cannot delete them.

The following example shows the definition of the Basic functionality level:

```
<Level>
  <Name>Basic</Name>
  <Role>All</Role>
  <FeatureID>Jobs</FeatureID>
  <FeatureID>Documents</FeatureID>
  <FeatureID>Channels</FeatureID>
  <SubfeatureID>DeleteFile</SubfeatureID>
  <SubfeatureID>InteractiveViewing</SubfeatureID>
</Level>
```

Every functionality level entry in the configuration file must have the five components shown in the following sections.

Name

Use a unique alphanumeric string for the functionality level name, enclosed within the <Name> and </Name> tags, such as <Name>Intermediate</Name>.

Role

The Role component defines the name of the Encyclopedia volume security role that corresponds to the functionality level. Both the security role and the functionality level must exist before you can assign the functionality level to a user. Enclose the security role name within <Role> and </Role> tags, such as <Role>Active Portal Intermediate</Role>.

Features

There are five features, which are described in Table 3-4.

Table 3-4 Features of functionality levels

Feature	Description
Channels	Provides access to channels
Customization	Provides access to skin customization
Documents	Provides access to files and folders
Jobs	Supports submitting and accessing jobs
Mobile	Provides access to BIRT mobile viewing
Search	Provides access to file and folder search

Enclose the feature within <FeatureID> and </FeatureID> tags. When you omit a feature from a functionality level, the corresponding side menu or banner item is not visible to anyone using that functionality level. For example, the Search feature is not available to the Basic functionality level, so the Search link does not appear in the banner for a user at the Basic functionality level.

Feature IDs

Functionality-level.config defines the features that are available to Information Console users as well as functionality levels. The following example shows the Documents feature definition from functionality-level.config:

```
<Feature>
  <ID>Documents</ID><Labelkey>SBAR_DOCUMENTS</Labelkey><Link>
    /getfolderitems.do</Link>
  <SmallIcon>/portal/activePortal/images/
    filesfoldersicon16x16.gif
  </SmallIcon>
  <LargeIcon>/portal/activePortal/images/filesfoldersicon.gif
  </LargeIcon>
</Feature>
```

The ID identifies the feature for Information Console. The label key appears on the side menu for Documents, Jobs, and Channels, or in the banner for Search and Customization. The link specifies the action that is executed for the feature. The small and large icons represent the feature in the side menu. Only the side menu features use the small and large icons.

Although you can customize the labels and links of all five features, do not change the <ID> or <Labelkey> tag values. Information Console uses these tags to identify the features and perform resource management. The Labelkey provides the resource to use for the feature's text label.

Changing the Link tag's value specifies a different action to execute. Changing the icon files changes the side menu's appearance. The small icons are used by the Treeview skin and are 16x16 pixels. The large icons are used by the Classic skin and are 32x32 pixels. The Tabbed skin does not use icons. Link and icon file names are relative to <context root>.

Subfeatures

A subfeature corresponds to an action you can perform using the Information Console user interface. A user must have appropriate privileges to create, delete, or share files or folders. Table 3-5 describes the subfeatures.

Table 3-5 Subfeatures of the features described in Table 3-4

Feature	Subfeature	Supported functionality
Channels	SubscribeChannel	Subscribing to channels.
Documents	AddFile	Uploading files.
Documents	CreateFolder	Creating folders.
Documents	DeleteFile	Deleting files.
Documents	DeleteFolder	Deleting folders.
Documents	DownloadFile	Downloading files.
Documents	ShareFile	Sharing files.
Jobs	JobPriority	Setting job priority, up to the user’s maximum job priority.
Jobs	SelfNotification WithAttachment	E-mail notification for successful jobs.
None	InteractiveViewing	Using BIRT Interactive Viewer.
None	AdvancedData	Used in BIRT Studio.
None	DashboardBusiness User	Viewing and editing dashboards and gadgets.
None	DashboardDeveloper	Creating and configuring gadgets and dashboards.
None	ShareDashboard	Sharing dashboards. Requires either DashboardBusinessUser or DashboardDeveloper.

Specify one subfeature to a line and enclose each subfeature within <SubfeatureID> and </SubfeatureID> tags. Each subfeature is associated with a feature. You cannot include a subfeature in a functionality level if its corresponding feature is not available to that functionality level.

Configuring Information Console locales

<context root>\WEB-INF\localemap.xml contains the locales available to Information Console. Add locales to this file using the same format as the existing locales. To see each locale in the file, search for one of the following strings:

```
<Locale  
or:  
<DisplayName>
```

Searching for <Locale places the cursor on the line having the ID for the locale. Searching for <DisplayName> places the cursor on the line having the descriptive name for the locale.

Typically, the locale names have the following syntax:

```
<language>_<country>
```

For example, ar_EG is Arabic (Egypt). A language spoken in multiple countries has multiple locale names for which the language code is the same and the country code has several values. For example, en_US is the locale for English (United States), en_AU is the locale for English (Australia), and en_BZ is the locale for English (Belize). Some countries have several locales, one for each language. For example, Canada has both en_CA for English (Canada) and fr_CA for French (Canada). You specify a default locale for a custom web application in <context root>\WEB-INF\web.xml.

Configuring Information Console time zones

<context root>\WEB-INF\TimeZones.xml contains the time zones available to Information Console. Add time zones to this file by using the same format as the existing time zones. To see each time zone in the file, search for one of the following strings:

```
<TimeZone
```

or:

```
<DisplayName>
```

Searching for <TimeZone places the cursor on the line having the ID for the time zone. Searching for <DisplayName> places the cursor on the line having the descriptive name for the time zone.

Some time zone names have short abbreviations for the ID. All time zone names have a full descriptive ID, such as Samoa Standard Time or Greenwich Standard Time. The DisplayName provides the relative time from Greenwich Standard Time and one or more locations that the time zone includes. You specify a default time zone for a custom web application in <context root>\WEB-INF\web.xml.

Customizing messages and text according to locale

Error messages and text for Information Console are encoded in resource files compressed in the <context root>/WEB-INF/lib/resources.jar file. The properties files contain entries for the interface text and error codes Information Console generates.

For reference, the <context root>/WEB-INF/ErrorMessage.txt file lists the default error codes used by Information Console. The \com\actuate\reportcast\resources\ErrorMessages.properties file within the resources.jar archive contains error messages for the default locale. Information Console uses messages

from this file if no locale-specific message for the error exists. Not all of the codes exist in the default `ErrorMessages.properties` because iHub directly generates many of them in the SOAP messages sent to Information Console.

Override iHub and Information Console messages using a locale-specific error messages file. In addition to the default `ErrorMessages.properties` file, Information Console provides several localized error message files, such as `ErrorMessages_de_DE.properties`. This file contains the German language messages for the Germany locale. To specify error messages to a certain locale, modify the existing error message file for that locale or create a new file for the locale. By convention, the format of a locale-specific error message file name includes the language and locale codes at the end of the file name separated by underscore characters.

For example:

```
ErrorMessages_de_DE.properties
```

- `de` is the language code for German.
- `DE` is the Germany country code.

These values for language and locale codes are defined in `localemap.xml`.

Because alphabets for different languages are dissimilar and Information Console uses ASCII encoding for these files, you must convert new or edited files into ASCII format. To convert the files to ASCII, modify the properties file using an editor that saves to the UTF-8 format and convert the file to ASCII using the Java `native2ascii` utility using the `-encoding UTF-8` switch. The `native2ascii` utility installs with any Java Developer Kit in the `<JDK home>/bin` directory. Model the format of new messages after those in the `ErrorMessages.properties` file.

When your modifications are complete, recompress the `resources.jar` archive using the Java `jar` utility, retaining the original directory structure for the archive. Copy the new `resources.jar` file to the `<context root>/WEB-INF/lib` directory, restart the Actuate 11 Apache Tomcat for Information Console service, and log in using the locale for the modified messages file. Confirm that the new messages file was loaded by examining the error messages generated by Information Console using that specific locale.

Error messages appear in pop-up windows when an error is encountered. The window is an operating system window, not an HTML frame. If you use a language-specific version of Windows corresponding to the locale you are viewing, the localized message shows up correctly. If you have not loaded the Windows language pack for a language, the text of a message appears as empty squares instead of text.

Configuring Shindig 2.0 for a WAR or EAR deployment

To enable shindig 2.0 support for a deployable Information Console's WAR or EAR file, modify the host name, port, and context root in the following configuration files:

```
<context root>/WEB-INF/web.xml
<context root>/WEB-INF/classes/shindig.properties
<context root>/WEB-INF/classes/containers/default/container.js
```

In web.xml, update the following parameters:

```
<param-name>system.properties</param-name>
<param-value>
    shindig.host=<host name>
    shindig.port=<port>
</param-value>
```

- <host name> is the name of the web server hosting the Information Console web application.
- <port> is the TCP port assigned to the Information Console web application.

In shindig.config, update the following parameter:

```
shindig.signing.global-callback-url=http://<host name>:<port>/
<context>/gadgets/oauthcallback
```

- <host name> is the name of the web server hosting the Information Console web application.
- <port> is the TCP port assigned to the Information Console web application.
- <context> is the context root of the Information Console web application.

In container.js update the following parameters:

```
"gadgets.jsUriTemplate" : "http://%host%/<context>/gadgets/js/
%js%"
"gadgets.oauthGadgetCallbackTemplate" : "://%host%/<context>/
gadgets/oauthcallback"
"gadgets.osDataUri" : "http://%host%/<context>/social/rpc"
"proxyUrl" : "://%host%/<context>/gadgets/
proxy?refresh=%refresh%&url=%url%",
"jsonProxyUrl" : "://%host%/<context>/gadgets/makeRequest"
"path" : "http://%host%/<context>/social"
"endPoints" : [ "http://%host%/<context>/social/rpc", "http://
%host%/<context>/gadgets/api/rpc" ]
```

<context> is the context root of the Information Console web application.

Configuring the connection to iHub

Information Console provides the ability to connect to iHub, an Encyclopedia volume, and manage reports on remote systems. Configure the repository, network, and Message Distribution service for Information Console using parameters in web.xml. These parameters control Information Console's connection to iHub and the Encyclopedia volume. Table 3-6 describes the configuration parameters for networking with iHub.

Table 3-6 iHub connection web.xml parameters

Parameter name	Description
AUTO_SAVE_DASHBOARD_DELAY	Controls how long, in seconds, the dashboard engine should wait before sending a save message to persist the personal dashboard file. To disable auto save, set this value to 0.
DASHBOARD_SHARED_RESOURCES	Specifies the path for the shared dashboard and gadget resources on the Encyclopedia volume. The gadget gallery displays the contents of this folder under the shared folder and is the default location when sharing dashboards.
MAX_CONNECTIONS_PER_SERVER	Indicates the maximum number of Actuate Information Console connections to Actuate BIRT iHub. Actuate pools connections to increase efficiency. Choose a number of connections that satisfies the most requests concurrently without requiring an unreasonable amount of memory. Begin with a value equal to the number of threads available in your application server. The value for this parameter must be greater than 0. The default value is 150.
MDS_ENABLED	Indicates whether to enable the Message Distribution service. The default value is True, which enables the Message Distribution service. For more information about the Message Distribution service, see "Understanding Actuate Information Console load balancing" in Chapter 1, "Introducing Actuate Information Console."
MDS_REFRESH_FREQUENCY_SECONDS	Indicates, in seconds, how quickly Actuate Information Console detects an offline or new node in a cluster. If MDS_ENABLED is True, Information Console refreshes the list of available nodes from Actuate BIRT iHub at the time interval specified. The default value is 300 seconds.
REPOSITORY_CACHE_TIMEOUT_SEC	Specifies how long a repository cache remains valid. When the cache becomes invalid, any user actions refresh the cache for the time-out duration. The default value is 900 seconds.

(continues)

Table 3-6 iHub connection web.xml parameters (continued)

Parameter name	Description
TEMP_FOLDER_LOCATION	Specifies the directory Information Console uses to temporarily store files from an Encyclopedia volume if viewing the file requires a location on the web server. If the value is not an absolute directory path name, Actuate Information Console locates the directory in the Information Console home directory. The default value is temp in the Information Console home directory. The Information Console user must have write permission for the directory. When deploying more than one context root or separate server, set a unique path for each.
VOLUME_PROFILE_LOCATION	Path to the volume profile configuration file from the context root. Default value is /WEB-INF/VolumeProfile.xml.

Configuring the BIRT Viewer and Interactive Viewer

The BIRT Viewer provides the ability to view a BIRT report. The Interactive Viewer supports modifying many aspects of the report's layout and formatting. These viewers are available in Information Console with the appropriate licensed iHub system option. They are also available as Java Components. Parameters in web.xml configure these viewers. For information on those configuration parameters, see *Working with Actuate BIRT Viewers*.

Configuring BIRT Studio

BIRT Studio is a report design tool that you use to design BIRT reports. This designer is available in Information Console with the appropriate licensed iHub system option. It is also available as a Java Component. Parameters in web.xml configure it. For information on those configuration parameters, see *Using BIRT Studio - iHub Edition*.

Configuring BIRT Data Analyzer

BIRT Data Analyzer extends the functionality of BIRT Interactive Viewer to perform analytics on a cross tab. You can configure performance enhancements for Data Analyzer in web.xml. For information on those configuration parameters, see *Using BIRT Data Analyzer*.

Actuate Information Console URIs

This chapter contains the following topics:

- Actuate Information Console URIs overview
- Actuate Information Console URIs quick reference
- Common URI parameters
- Information Console Struts actions
- Actuate Information Console URIs reference
- Actuate BIRT Viewer URIs reference

Actuate Information Console URIs overview

This chapter describes Actuate Information Console URIs. Information Console JSPs manage content. The following sections provide quick reference tables and detailed reference information about Actuate Information Console URIs. An Actuate Information Console URI is a directive to Actuate Information Console to perform an action, such as showing a list of files.

Information Console pages use the .do extension for Struts action mapping to a page. The complete page name appears as part of the reference material. Actuate Information Console page and folder names are case-sensitive.

Information Console supports two viewers, which have specific URLs associated with them. The detailed reference material for Information Console and the viewers is divided into the following categories:

- Actuate Information Console URIs reference
- Actuate BIRT Viewer URIs reference

Actuate Information Console URIs quick reference

Table 4-1 lists the Actuate Information Console URIs. For more information about the Information Console directory structure, see “Understanding Information Console directory structure” in Chapter 2, “Creating a custom Information Console web application.”

Table 4-1 Actuate Information Console URI pages

Actuate Information Console page	Description
about page	Displays information about Actuate Information Console.
banner page	Displays a banner at the top of each Actuate Information Console page.
browse file page	Provides file and folder browsing functionality for the submit request pages.
browse page	See browse file page.
calendar page	Provides calendar functionality for submit request’s scheduling feature.
canceljob page	See request drop page.
channels page	Displays the channels property sheet.
completed request page	Lists all completed requests.

Table 4-1 Actuate Information Console URI pages (continued)

Actuate Information Console page	Description
create folder page	Creates a folder.
delete file status page	Displays whether a file was successfully deleted.
dashboard page	Provides the dashboard interface.
delete job page	Deletes a scheduled job.
delete status page	Deletes the completed job notice.
detail page	Supports error handling and presenting object details.
do_update page	See options page.
drop page	Supports deleting files or canceling running jobs.
error page	Retrieves an error message from the exception or the request and displays it.
execute report page	Submits a run report job request to the server.
general options page	Displays the general user settings and environment settings property sheet.
getfiledetails page	See file or folder detail page.
getfolderitems page	See file and folder index page.
getjobdetails page	See request detail page.
home page	Provides the link from the My Folder button to the Actuate Information Console home page.
list page	Supports listing channels, channel contents, and Encyclopedia objects.
login banner page	Provides the banner for the Actuate Information Console login page.
login page	Logs into the reporting web application.
logout page	Logs the user out of the current session and clears all user settings, such as filters.
notification page	Supports specifying the current user's request notification options.
options page	Updates options and user settings. See also options index page.
output page	Presents a form to specify output information for report jobs, such as report object name and location.
page not found page	Displays an error message when a JSP is unavailable in Information Console.
parameters page	Presents a list of the request parameters.
pending page	Lists all requests awaiting execution.

(continues)

Table 4-1 Actuate Information Console URI pages (continued)

Actuate Information Console page	Description
ping page	Diagnostics for Actuate BIRT iHub System components.
privileges page	Prints report documents in PDF format.
privileges page	Sets file and folder privileges.
running page	Lists all requests currently executing.
schedule page	Presents a form for specifying scheduled report job request properties, such as date, time, recurring request, and immediate report job run.
scheduled job page	Lists all requests awaiting execution at specified dates and times.
search folders page	Searches folders recursively for files and folders.
selectjobs page	See requests index page.
submit job page	Submits a scheduled job request to the server.
viewer page for BIRT reports	Displays BIRT documents and the toolbar.

Common URI parameters

All Actuate Information Console URIs have the parameters shown in Table 4-2. String values that are too long are truncated for all parameters. The web browser that you use determines the length of parameters. The common URI parameters support Actuate Information Console authentication using cookies.

Table 4-2 Common Actuate Information Console URI parameters

URI parameter	Description
forceLogin	True to force a login, False to display the login page. The default is False. For example, when switching between Encyclopedia volumes and using an Information Console security manager class, set forceLogin=true to force the Information Console Login module to call the security manager to perform the login operation. The login operation is described in Chapter 9, "Using Actuate Information Console security."
iPortalID	The unique authentication ID assigned to the user upon successful login. Use this parameter in conjunction with the userID parameter to ensure that a user's personalized settings appear in the Information Console pages.
locale	The current user's locale, such as U.S. English (en-US).

Table 4-2 Common Actuate Information Console URI parameters

URI parameter	Description
password	The password associated with the userID.
serverURL	The URI that accesses the Actuate BIRT iHub, such as <code>http://Services:8000</code> .
timezone	The current user's time zone.
userID	The user's unique identifier, required to log in to the repository. Use this parameter in conjunction with the iPortalID parameter to ensure that a user's personalized settings appear in the Information Console pages.
volume	The volume to which the user connects. This parameter overrides the volume from the __vp parameter if they are used together.
__vp	The name of a server configured in VolumeProfile.xml. Information Console uses the volume information in a VolumeProfile entry except when a volume parameter specifies a different one.

The following Information Console URI shows most of the common URI parameters in use:

```
http://localhost:8700/iportal/dashboard?folder=/Training
&volume=Encyc2&locale=en_AU&userID=Mike&password=pw123
&serverURL=http://server1:8000&timeZone=Australia/Perth
```

This URI lists the contents of the Training folder in the Encyc2 Encyclopedia volume on the Actuate BIRT iHub named server1 at port 8000. The locale is set to Australian English and the time zone is Australia/Perth (GMT plus eight hours). The user is Mike and the password is pw123. The password is shown in plain text, as entered.

If the server and volume information for server1 above is configured as a Volume Profile, you can use a simplified URL as shown in the following lines:

```
http://localhost:8700/iportal/dashboard?folder=/Training
&__vp=server1&locale=en_AU&userID=Mike&password=pw123
&timeZone=Australia/Perth
```

Information Console Struts actions

The following tables summarizes the global forwards and actions defined in struts-config.xml.

Table 4-3 lists the global forwards defined in struts-config.xml.

Table 4-3 Actuate Information Console global forwards

Action	Forward
authexpired	/login.do
browsefile	/browsefile.do
canceljob	/canceljob.do
da	/da
dashboard	/dashboard
deletefile	/deletefile.do
deletejob	/deletejob.do
deletejobnotice	/deletejobnotice.do
deletejobschedule	/deletejobschedule.do
downloadfile	/servlet/DownloadFile
error	/private/common/errors/errorpage.jsp
errorportlet	/private/common/errors/errorportlet.jsp
executedocument	/executedocument.do
executereport	/executereport.do
getjobdetails	/getjobdetails.do
getrequesterjobdetails	/getrequesterjobdetails.do
getsavedsearch	/viewer/getsavedsearch.do
goto	/private/common/goto.jsp
iv	/iv
login	/login.do
logout	/logout.do
requestercanceljob	/requestercanceljob.do
requesterdeletejob	/requesterdeletejob.do
requesterdeletejob schedule	/requesterdeletejobshedule.do
skinerror	/private/common/errors/error.jsp
viewframeset	/viewer/viewframeset.jsp
viewpage	/servlet/ViewPage
wr	/wr

Table 4-4 lists the action, input JSP, and forward name and path defined in struts-config.xml.

Table 4-4 Actuate Information Console actions

Action	Input JSP	Forward name path
/browsefile	/iportal/activePortal /private/newrequest /browse.jsp	name=success path=/iportal/activePortal/private /newrequest/browse.jsp name=browseFile path=/iportal/activePortal/private /common/browseFile.jsp
/browseportletfile	/iportal/portlets /browsefile.jsp	name=success path=/iportal/portlets/browsefile.jsp
/canceljob		name=success path=/iportal/activePortal/private /jobs/joboperationstatus.jsp
/cancelreport		name=Succeeded path=/iportal/activePortal/viewer /closewindow.jsp name=Failed path=/iportal/activePortal/viewer /closewindow.jsp?status=failed name=InActive path=/iportal/activePortal/viewer /closewindow.jsp?status=inactive
/createfolder		name=success path=/getfolderitems.do name=cancel path=/getfolderitems.do name=showform path=/iportal/activePortal/private /filesfolders/createfolder.jsp
/customize		name=success path=/iportal/activePortal/private /customization/skinmanager.jsp name=downloadwar path=/servlet/CacheDownload

(continues)

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/deletefile		name=success path=/iportal/activePortal/private/filesfolders/deletefilestatus.jsp name=error path=/iportal/activePortal/private/filesfolders/deletefilestatus.jsp name=confirm path=/iportal/activePortal/private/filesfolders/confirm.jsp
/deletejob		name=success path=/iportal/activePortal/private/jobs/joboperationstatus.jsp
/deletejobnotice		name=success path=/iportal/activePortal/private/jobs/joboperationstatus.jsp
/deletejobschedule		name=success path=/iportal/activePortal/private/jobs/joboperationstatus.jsp
/editTableRow	/iportal/activePortal/private/parameters/table/roweditor.jsp	name=close path=/iportal/activePortal/private/parameters/table/close.jsp name=tableRowEditor path=/iportal/activePortal/private/parameters/table/roweditor.jsp
/executedocument		name=success path=/executereport.do
/executereport	/private/newrequest/newrequest.jsp	name=viewbirt path=/iv name=viewreport path=/servlet/DownloadFile name=wait path=/iportal/activePortal/private/newrequest/waitforexecution.jsp
/filefoldersprivilege	/iportal/activePortal/private/filesfolders/privilege.jsp	name=success path=/getfolderitems.do
/getfiledetails		name=success path=/iportal/activePortal/private/filesfolders/filedetail.jsp

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/getfolderitems		name=success path=/iportal/activePortal/private/filesfolders/filefolderlist.jsp name=dashboard path=/dashboard
/getjobdetails		name=success path=/iportal/activePortal/private/jobs/getjobdetails.jsp
/getnoticejobdetails		name=success path=/iportal/activePortal/private/jobs/getjobdetails.jsp
/getportletfolder items		name=success path=/iportal/portlets/filefolderlist/filefolderlistportlet.jsp
/getrequesterjob details		name=success path=/iportal/activePortal/private/jobs/getrequesterjobdetails.jsp
/iPortalLogin	/iportal/login.jsp	name=landing path=/landing.jsp name=iPortalLoginForm path=/login.jsp
/iv	/iportal/activePortal/private/newrequest/newrequest.jsp	name=iv path=/iv name=viewbirt path=/iv
/login	/iportal/activePortal/private/login.jsp	name=loginform path=/iportal/activePortal/private/login.jsp name=success path=/getfolderitems.do name=dashboard path=/dashboard name=ajclogin path=/ajclanding.jsp name=landing path=/landing.jsp

(continues)

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/logout		name=landing path=/landing.jsp
/options	/iportal/activePortal /private/options /options.jsp	name=success path=/iportal/activePortal/private /options/options.jsp name=saved path=/getfolderitems.do name=dashboard path=/iportal/activePortal/private /options/options.jsp
/options/save	/iportal/activePortal /private/options /options.jsp	name=success path=/getfolderitems.do name=saved path=/getfolderitems.do
/ping		name=success path=/iportal/activePortal/private /diagnosis/pingresponse.jsp
/requestercanceljob		name=success path=/iportal/activePortal/private /jobs/requesterjoboperationstatus.jsp
/requesterdeletejob		name=success path=/iportal/activePortal/private /jobs/requesterjoboperationstatus.jsp
/requesterdeletejob schedule		name=success path=/iportal/activePortal/private /jobs/requesterjoboperationstatus.jsp
/searchfiles		name=success path="/iportal/activePortal/private /filesfolders/search/filefolderlist.jsp
/selectchannels		name=channellist path=/iportal/activePortal/private /channels/channellist.jsp
/selectjobnotices		name=success path=/iportal/activePortal/private /channels/channelnoticelist.jsp
/selectjobs		name=success path=/iportal/activePortal/private /jobs/selectjobs.jsp

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/searchfiles		name=success path=/iportal/activePortal/private/filesfolders/search/filefolderlist.jsp
/skinedit	/customize.do	name=success path=/iportal/activePortal/private/customization/skinedit.jsp
/submitjob	/iportal/activePortal/private/newrequest/newrequest.jsp	name=success path=/iportal/activePortal/private/newrequest/submitjobstatus.jsp name=viewreport path=/servlet/DownloadFile
/subscribeChannel	/iportal/activePortal/private/channels/channelssubscribe.jsp	name=success path=/selectchannels.do
/tableList	/iportal/activePortal/private/parameters/table/tableparameters.jsp	name=close path=/iportal/activePortal/private/parameters/table/close.jsp name=tableParamList path=/iportal/activePortal/private/parameters/table/tableparameters.jsp
/treebrowser		name=success path=/iportal/activePortal/private/filesfolders/treebrowser.jsp
/updatechannel		name=success path=/iportal/activePortal/private/channels/channeloperationstatus.jsp
/uploadimage		name=success path=/iportal/activePortal/private/customization/fileupload.jsp
/uploadlicense		name=success path=/iportal/admin/fileupload.js

(continues)

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/viewer /getsavedsearch		name=success path=/getfolderitems.do name=searchreport path=/iportal/activePortal/viewer/searchreportpage.jsp name=requestsearch path=/iportal/activePortal/viewer/requestsearch.jsp
/viewer/savesearch	/iportal/activePortal/viewer/savesearch.jsp	name=success path=/iportal/activePortal/viewer/requestsearch.jsp name=browse path=/browsefile.do
/waitforreport execution	/iportal/activePortal/private/newrequest/waitforexecution.jsp	name=success path=/iportal/activePortal/viewer/viewreport.jsp name=fail path=/iportal/activePortal/viewer/closewindow.jsp

Actuate Information Console URIs reference

This section provides the detailed reference for Actuate Information Console URIs. In the definitions, <context root> represents the name of your Actuate Information Console context root, initially iportal. Table 4-5 lists the topics this chapter covers and the file names discussed in each topic. All pages are under the Information Console context root.

Table 4-5 Actuate Information Console pages

Topic	Information Console file
about page	iportal\activePortal\private\options\about.jsp
banner page	iportal\activePortal\private\common\banner.jsp
browse file page	browsefile.do
calendar page	iportal\activePortal\private\newrequest\calendar.jsp
channels page	iportal\activePortal\private\options\channels.js

Table 4-5 Actuate Information Console pages (continued)

Topic	Information Console file
completed request page	iportal\activePortal\private\jobs\completedjob.jsp
create folder page	createfolder.do
dashboard page	DashboardServlet
delete file status page	iportal\activePortal\private\filesfolders\deletefilestatus.jsp
delete job page	deletejob.do
delete status page	deletejobnotice.do
detail page	
■ error detail page	iportal\activePortal\errors\detail.jsp getfiledetails.do
■ file or folder detail page	iportal\activePortal\private\filesfolders\filedetail.jsp
■ request detail page	getjobdetails.do iportal\activePortal\private\jobs\getjobdetails.jsp
drop page	
■ file or folder drop page	deletefile.do
■ request drop page	canceljob.do
error page	errors\error.jsp iportal\activePortal\private\common\errors\error.jsp
execute report page	executereport.do
general options page	iportal\activePortal\private\options\general.jsp
home page	iportal\activePortal\private\common\breadcrumb.jsp
index page	
■ file and folder index page	getfolderitems.do iportal\activePortal\private\filesfolders\filefolderlist.jsp
■ new request index page	executereport.do

(continues)

Table 4-5 Actuate Information Console pages (continued)

Topic	Information Console file
■ options index page	options.do iportal\activePortal\private\options\options.jsp
■ requests index page	selectjobs.do iportal\activePortal\private\jobs\selectjobs.jsp
list pages	
■ channels list page	selectchannels.do iportal\activePortal\private\channels\channellist.jsp
■ channel contents list page	iportal\activePortal\private\channels\channelnoticelist.jsp
■ file and folder list page	getfolderitems.do iportal\activePortal\private\filesfolders\filefolderlist.jsp
login banner page	iportal\activePortal\private\login_banner.jsp
login page	login.do iportal\activePortal\private\login.jsp
logout page	logout.do
notification page	iportal\activePortal\private\options\notification.jsp
options page	options.do iportal\activePortal\private\options\options.jsp
output page	iportal\activePortal\private\newrequest\output.jsp
page not found page	iportal\activePortal\errors\pagenotfound.jsp
parameters page	iportal\activePortal\private\newrequest\parameters.jsp
pending page	iportal\activePortal\private\jobs\pendingjob.jsp
ping page	ping.do iportal\activePortal\private\diagnosis\pingresponse.jsp
privileges page	iportal\activePortal\viewer\print.jsp
running page	iportal\activePortal\private\jobs\runningjob.jsp
schedule page	iportal\activePortal\private\newrequest\schedule.jsp
scheduled job page	iportal\activePortal\private\jobs\scheduledjob.jsp

Table 4-5 Actuate Information Console pages (continued)

Topic	Information Console file
search folders page	searchfiles.do iportal\activePortal\private\filesfolders\search\filefolderlist.jsp
submit job page	submitjob.do iportal\activePortal\private\newrequest\submitjobstatus.jsp
viewer page for BIRT reports	IVServlet

about page

Displays the About page, containing information about Actuate Information Console. Called when the user chooses the About tab on the Options page.

The default About page for Information Console is similar to Figure 4-1.

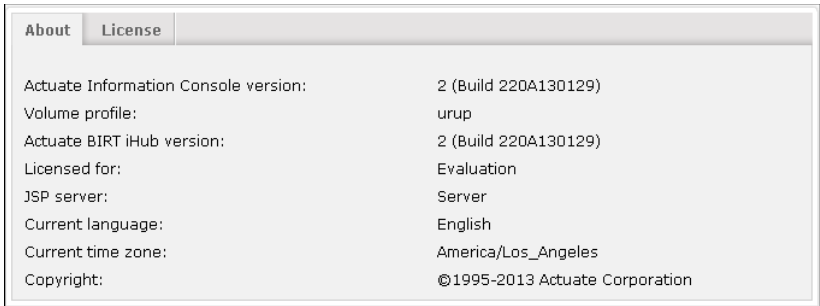


Figure 4-1 Information Console About page

Name	<context root>iportal\activePortal\private\options\about.jsp
Parameters	The about page uses the common URI parameters.
Used by	iportal\activePortal\private\options\optionspage.jsp

banner page

Provides the banner that appears across the top of all Actuate Information Console web pages. The default banner displays the Actuate logo, user name, and license, and provides links for Logout, Options, and Help. The banner page obtains the user name from variables maintained by the authenticate page.

Name	<context root>\portal\activePortal\private\common\banner.jsp
Used by	portal\activePortal\private\login.jsp portal\activePortal\private\channels\channelnoticelist.jsp portal\activePortal\private\channels\channeloperationstatus.jsp portal\activePortal\private\filesfolders\deletefilestatus.jsp portal\activePortal\private\filesfolders\filedetail.jsp portal\activePortal\private\filesfolders\filefolderlist.jsp portal\activePortal\private\jobs\getjobdetails.jsp portal\activePortal\private\jobs\joboperationstatus.jsp portal\activePortal\private\jobs\selectjobs.jsp portal\activePortal\private\newrequest\newrequest.jsp portal\activePortal\private\newrequest\newrequest2.jsp portal\activePortal\private\newrequest\submitjobstatus.jsp portal\activePortal\private\options\options.jsp

browse file page

Contains file and folder browsing functionality used by submit request pages.

Name	<context root>\browsefile.do
Parameters	workingFolder is the name of the folder for which to display contents in the folder browser window. The browse file page also uses the common URI parameters.
Used by	portal\activePortal\private\newrequest\browse.jsp

calendar page

Provides calendar functionality for the submit request scheduling feature.

Name	<context root>\portal\activePortal\private\newrequest\calendar.jsp
Used by	portal\activePortal\private\newrequest\newrequestpage.jsp

channels page

Displays the channels property sheet. The channels page presents a list of all channels available on the current volume. Channels to which the user subscribes appear with their check boxes selected.

The channels page looks like Figure 4-2.

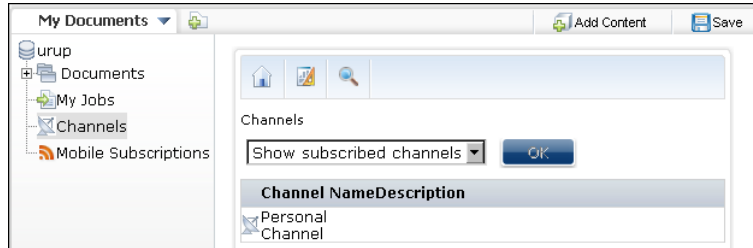


Figure 4-2 Channels page

Users choose which channels they want to see in the list by specifying a filter. For example, to see all Marketing Communications channels, the user might type the filter Mar* in the Filter field. Channels uses the HTTP session variable AcChannelFilter to save the current filter value. AcChannelFilter works if cookies are enabled. For more information, see *Managing an Encyclopedia Volume*.

Name <context root>\portal\activePortal\private\options\channels.jsp
Used by iportal\activePortal\private\options\optionspage.jsp

completed request page

Lists all completed requests. The completed request page lists all report jobs that have executed and are available or whose execution failed.

The completed request page looks like Figure 4-3.

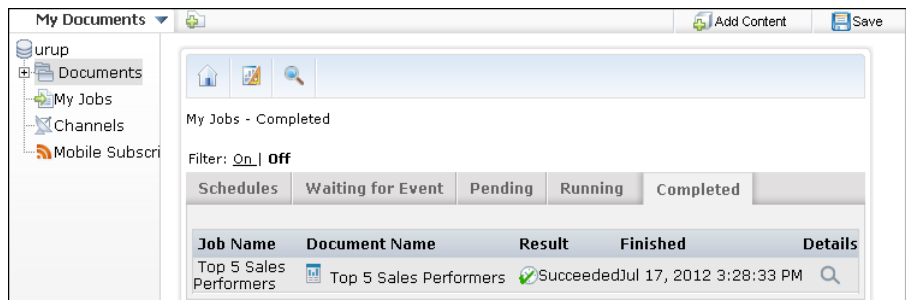


Figure 4-3 Completed request page

Name <context root>\portal\activePortal\private\jobs\completedjob.jsp
Parameters The completed request page uses the common URI parameters.
Used by iportal\activePortal\private\jobs\selectjobscontent.jsp

create folder page

Creates a folder in the current Encyclopedia volume. Createfolder.do uses <context root>\portal\activePortal\private\filesfolders\createfolder.jsp to create the new folder.

Name <context root>\createfolder.do

Parameters Table 4-6 lists and describes the parameters for the create folder page. The create folder page also uses the common URI parameters.

Table 4-6 Parameters for create folder URI

URI parameter	Description
workingFolderID	The ID of the folder to contain the new folder. Specify either workingFolderID or workingFolderName.
workingFolderName	The name of the folder to contain the new folder. Specify either workingFolderID or workingFolderName.

Used by Not applicable.

dashboard page

Provides the dashboard interface servlet for Information Console.

Name <context root>\dashboard

Used by Not applicable.

delete file status page

Summarizes the result of a deletion performed by the drop page and indicates whether a file was successfully deleted. The delete file status page includes authenticate to obtain user session data. Information Console performs the deletion as part of an action and then forwards to the delete file status page.

Name <context root>\portal\activePortal\private\filesfolders\deletefilestatus.jsp

Used by Not applicable.

delete job page

Deletes the specified job, then redirects the page to a completion status page. Specify the name or the ID of the job to delete.

The default redirection JSP is <context root>\iportal\activePortal\private\jobs\joboperationstatus.jsp.

Name <context root>\deletejob.do

Parameters Table 4-7 lists and describes the parameters for the delete job page. The delete job page also uses the common URI parameters.

Table 4-7 Parameters for delete job URI

URI parameter	Description
jobID	Unique request identifier.
jobName	The name of the job to delete. This parameter is ignored if jobID is also specified.
jobState	The state of the job to delete.
redirect	URI to which to redirect the job deletion page.

Used by Not applicable.

delete status page

Deletes a job notice corresponding to a request. Specify the job notice by name or by ID.

Name <context root>\deletejobnotice.do

The default redirection action forwards to <context root>\iportal\activePortal\private\jobs\joboperationstatus.jsp.

Parameters Table 4-8 lists and describes the parameters for the delete status page. The delete status page also uses the common URI parameters.

Table 4-8 Parameters for delete status URI

URI parameter	Description
channelID	The unique identifier of the channel to delete the job notice from.
channelName	The name of the channel to delete the job notice from.
jobID	Unique request identifier.

(continues)

Table 4-8 Parameters for delete status URI (continued)

URI parameter	Description
jobName	The name of the job notice to delete. This parameter is ignored if jobID is also specified.
jobState	The state of the job to delete.
redirect	URL to which to redirect the delete status page.
userName	The name of the user to notify about the deleted job.

Used by Not applicable.

detail page

Displays detailed information about Encyclopedia volume objects. There are three detail pages:

- <context root>\iportal\activePortal\errors
- <context root>\iportal\activePortal\filesfolders
- <context root>\iportal\activePortal\requests

error detail page

Provides a template error page that can be embedded in another page.

Name <context root>\iportal\activePortal\errors\detail.jsp

Used by iportal\activePortal\private\common\errors\error.jsp

file or folder detail page

Displays detailed information about the selected viewable folder or file. Users request file or folder details by choosing the magnifying glass icon to the right of files or folders listed on the Encyclopedia folder page or breadcrumb. Users can request another document or delete the current file or folder from the file or folder detail page. filedetail.jsp uses the HTML code in <context root>\iportal\activePortal\private\filesfolders\filedetailcontent.jsp to display the information.

Name <context root>\getfiledetails.do

<context root>\iportal\activePortal\private\filesfolders\filedetail.jsp

Parameters Table 4-9 lists and describes the parameters for the file or folder detail page. The file or folder detail page also uses the common URI parameters.

Table 4-9 Parameters for file or folder detail URI

URI parameter	Description
name	The full path name of the Encyclopedia object for which to show details, if objectID is not specified.
objectID	The Encyclopedia object’s unique identifier.
version	The Encyclopedia object’s version number. The default is the latest version.

Used by Not applicable.

request detail page

Lists detailed request information for a specified job, as shown in Figure 4-4.

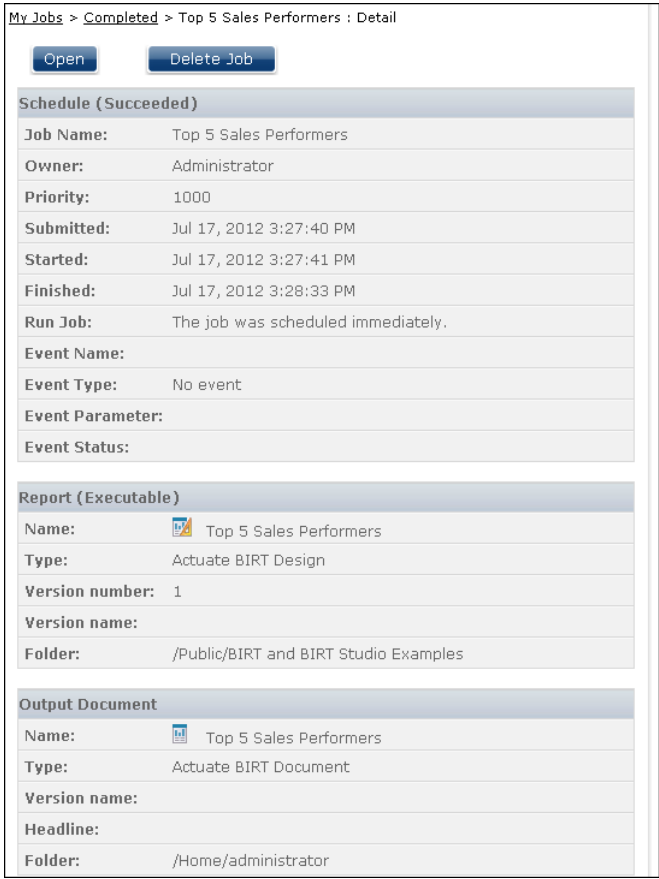


Figure 4-4 Request detail page

getjobdetails.jsp uses the HTML code in <context root>\iportal\activePortal\private\jobs\getjobdetailscontent.jsp to display the information.

Name <context root>\getjobdetails.do

<context root>\iportal\activePortal\private\jobs\getjobdetails.jsp

Parameters The request detail page uses the common URI parameters, as shown in Table 4-10.

Table 4-10 Parameters for request detail URI

URI parameter	Description
jobID	The job's unique identifier
userName	The user that submitted the job
channelName	The channel to receive the request

Used by iportal\activePortal\private\jobs\completedjob.jsp
iportal\activePortal\private\jobs\pendingjob.jsp
iportal\activePortal\private\jobs\runningjob.jsp
iportal\activePortal\private\jobs\scheduledjob.jsp

drop page

Deletes one or more files or folders, or cancels a running job.

file or folder drop page

Deletes the specified file or folder.

Name <context root>\deletefile.do

Parameters Table 4-11 lists and describes the parameters for the file or folder drop page. The file or folder drop page also uses the common URI parameters.

Table 4-11 Parameters for file or folder drop URI

URI parameter	Description
ID	The unique identifier of the object to delete.
name	The full path name of the Encyclopedia object to delete. Multiple name parameters, to delete more than one file or folder at a time, are allowed. This parameter is ignored if ID is also specified.
redirect	URI to navigate to upon completion. The default redirect page is processedaction_status.

Used by Not applicable.

request drop page

Cancels a running job.

Name <context root>\canceljob.do

Parameters Table 4-12 lists and describes the parameters for the request drop page. The request drop page also uses the common URI parameters.

Table 4-12 Parameters for request drop URI

URI parameter	Description
jobID	The unique identifier of the Encyclopedia object to delete.
jobName	The full path name of the Encyclopedia object to delete. This parameter is ignored if jobID is also specified.
jobState	The state of the job to delete. processedaction_status uses jobState to build a link to pass to the list of scheduled and completed jobs.
redirect	URI to navigate to upon completion. The default redirect page is processedaction_status.

Used by Not applicable.

error page

Displays the specified error message. Information Console uses two pages. All iHub Information Console code uses <context root>\iportal\activePortal\private\common\errors\error.jsp.

Name <context root>\iportal\activePortal\errors\error.jsp
<context root>\iportal\activePortal\private\common\errors\error.jsp

Used by iportal\activePortal\private\login.jsp
iportal\activePortal\private\common\closewindow.jsp
iportal\activePortal\private\common\sidebar.jsp
iportal\activePortal\private\common\errors\errorpage.jsp
iportal\activePortal\private\options\options.jsp
iportal\activePortal\private\templates\template.jsp

execute report page

Submits a run report job request to the Actuate BIRT iHub. When executing a report job, a Cancel button appears after a specified wait time passes. Change the time by setting the EXECUTE_REPORT_WAIT_TIME configuration parameter in

the appropriate Actuate Information Console configuration file. For reports that accept run-time parameters, you can set the parameter in the URL by adding an ampersand (&), the parameter name, and an equal (=) sign, followed by the parameter value in quotes.

Name <context root>\executereport.do

Parameters Table 4-13 lists and describes the parameters for the execute report page. The execute report page also uses the common URI parameters.

Table 4-13 Parameters for execute report URI

URI parameter	Description
__ageDays	Use with __ageHours to determine how long output objects exist before they are automatically deleted. Use only if __archivePolicy is set to Age. __ageDays can be any positive number.
__ageHours	Use with __ageDays to determine how long output objects exist before they are automatically deleted. Use only if __archivePolicy is set to Age. __ageHours can be any positive number.
__archiveBeforeDelete	Indicate whether to archive the output objects of the current request before deleting them, according to __archivePolicy's setting. Set this parameter to True to archive objects before deleting them. The default value is False. This parameter has no effect if __archivePolicy is set to Folder.
__archivePolicy	The archive policy to implement for the objects created as output for the current request. Values are folder, age, and date. Set to folder to use the archive policy that is already set for the folders to which the output is distributed. Set to age to delete objects older than a specific time period. Set to date to delete objects on a specific date.
__dateToDelete	The date on which to delete the output objects of the current request. Use only if __archivePolicy is set to Date. Set __dateToDelete to a date in a locale-specific format. The default format is mm/dd/yyyy.
__executableName	The name of the executable file for this request.
__headline	A descriptive tag line for a report document. Appears on Channel Contents. Use the character string %20 to represent a space in the headline string.
invokeSubmit	Controls whether the browser is redirected to the parameter screen or whether the report job is run immediately. If True, the report job is executed without displaying the parameters. If False, the parameters are displayed. False is the default.
__isnull	Sets the value of the named parameter to null. Use a parameter name as input.
__jobName	The name of the job to execute.

Table 4-13 Parameters for execute report URI (continued)

URI parameter	Description
<code>__limit</code>	Indicate whether to limit the number of versions of the output files for the current request. Set <code>__limit</code> to <code>Limit</code> to limit the number of versions. Any other value means that the number of versions is unlimited.
<code>__limitNumber</code>	The number of versions to which to limit the output files for the current request. Use only if <code>__limit</code> is set to <code>Limit</code> . <code>__limitNumber</code> can be any positive number.
<code>__outputFolderType</code>	Specifies the root of the output file name. Set to <code>Absolute</code> to use the full <code>__outputName</code> value starting from the Encyclopedia volume's root. Set to <code>Personal</code> to use the <code>__outputName</code> value relative to the user's home folder.
<code>__outputDocName</code>	Specifies a name for the output file.
<code>__overwrite</code>	New to create a new version of this report document, or <code>Old</code> to overwrite an existing report document. <code>New</code> is the default.
<code>__priority</code>	Specifies the job submission priority. Values are <code>High</code> , <code>Medium</code> , and <code>Low</code> .
<code>__priorityValue</code>	Specifies a number ranging from 1 to 1000 and corresponding to the job submission priority. Only specify values allowed by your functionality level.
<code>__progressive</code>	Indicates whether to display the report document after it generates. If <code>False</code> , the report document displays after it generates. If <code>True</code> , the report document displays progressively, as it generates.
<code>__recurringDay</code>	Specifies the scheduled recurring day on which to run the report job. Applies only to scheduled report jobs.
<code>__saveOutput</code>	Indicates whether to write the output document to the Encyclopedia volume. <code>True</code> saves the output in the Encyclopedia volume, applying the document archiving and file creation parameters. <code>False</code> does not save the output.
<code>__serverURL</code>	Contains the URI that accesses the JSP engine, such as <code>http://<i>Hub machine name>:8700</code> .
<code>__timeToDelete</code>	Specifies a time at which to delete an archived report document. Applies only scheduled report jobs.
<code>__users</code>	Contains the name of the user to notify of this scheduled request. You can notify more than one user. This parameter is valid only for scheduled jobs.

(continues)

Table 4-13 Parameters for execute report URI (continued)

URI parameter	Description
__versionName	Contains a string value for the new version name of this report document. The value can include a date/time expression enclosed in braces, {}, to ensure a unique version name.
__volume	Contains a string value specifying the volume for this report.
__wait	If "wait", Information Console waits for the report generation to be completed before displaying it. If "nowait", Information Console displays the first page right away even if the report job is not completed.

For example, the following URL executes the Sales By Territory.rptdesign report immediately with the Territory run-time parameter set to EMEA:

```
http://localhost:8700/iportal/executereport.do?  
__requesttype=immediate&__executableName=%2fPublic%2fBIRT and  
BIRT Studio Examples%2fSales by Territory.rptdesign&  
userid=Administrator&__saveOutput=false&Territory="EMEA"&  
invokeSubmit=True
```

Set string parameters to an empty string by adding the parameter to the executereport.do URI with no value following the equal (=) sign.

For example, the following line sets parameterA and parameterB to empty strings:

```
&parameterA=&parameterB=
```

The following parameter names are reserved for internal use only by the execute report page:

- doframe
- inputfile
- jobType
- name
- selectTab

Used by Not applicable.

general options page

Displays the general user settings and environment settings property sheet for the current user. There are two types of settings:

- User settings that apply only to this user:
 - Change password.
 - Change e-mail address.
- Environment settings that apply for all browsers on a single local machine:
 - Choose a skin to provide colors, fonts, images, and layout in the graphical user interface (GUI).
 - Choose a view to select a layout for the content area of pages providing lists of files and folders.
 - Enable and disable filter fields for Files and Folders, Channels, and Requests.
 - View documents in the current browser window or in a new browser window.

The general options page appears when the user chooses Options in the Actuate Information Console banner.

Name <context root>\portal\activePortal\private\options\general.jsp
Used by iportal\activePortal\private\options\optionspage.jsp

home page

Provides two sets of links. On the right side it provides a graphical and a text shortcut link from the My Folder button to the current user's Actuate Information Console home folder. If the Information Console installation supports BIRT Studio, there is another shortcut link, BIRT Studio, to the BIRT Studio. On the left side, it provides the links and other text for the breadcrumb, or path from the repository root to the current folder.

Users access their home page by choosing the My Folder link below the Actuate Information Console page banner. Figure 4-5 shows the default My Folder and breadcrumb links.

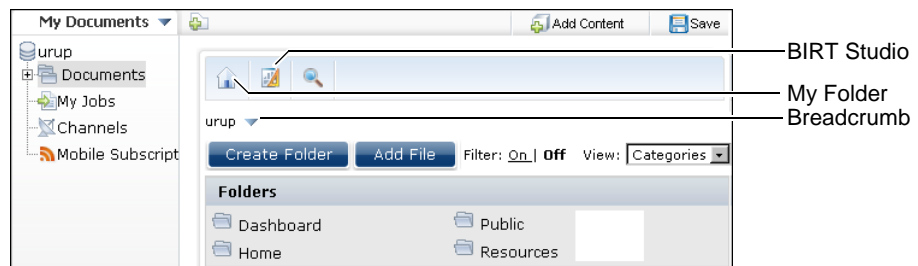


Figure 4-5 My Folder and breadcrumb links

Name <context root>\iportal\activePortal\private\common\breadcrumb.jsp

Used by iportal\activePortal\private\skins\tabbed\templates\mypagetabletemplate.jsp
iportal\activePortal\private\skins\tabbed\templates\template.jsp
iportal\activePortal\private\skins\classic\templates\template.jsp
iportal\activePortal\private\skins\treeview\templates\template.jsp

index page

Provides the entry point and structure for the parts of Actuate Information Console generated from multiple files.

file and folder index page

The default entry point to the Actuate Information Console web application. The file and folder index page provides the entry point and structure to support the Files and Folders functionality. The structure is a table that Actuate Information Console uses to format and present files and folders data. Page content varies depending on the Actuate Information Console directive.

The file and folder index page uses the banner page to provide the reporting web page banner. filefolderlist.jsp uses the HTML code in <context root>\iportal\activePortal\private\filesfolders\filefolderlistcontent.jsp to display files and folders data.

Name <context root>\getfolderitems.do

<context root>\iportal\activePortal\private\filesfolders\filefolderlist.jsp

Parameters Table 4-14 lists and describes the parameters for file and folder index page. The file and folder index page also uses the common URI parameters.

Table 4-14 Parameters for file and folder index URI

URI parameter	Description
startUpMessage	Specifies a message to appear when Actuate Information Console calls this page.
subpage	Specifies the content of the page. Possible values are: <ul style="list-style-type: none">■ _list: include list■ _detail: include detail Specifying any other value for subpage invokes the page not found page.

new request index page

Provides the entry point and structure to support the submit job functionality.

Name <context root>\executereport.do

Parameters Table 4-15 describes the parameter for the new request index page. The new request index page also uses the common URI parameters.

Table 4-15 Parameter for new request index URI

URI parameter	Description
homeFolder	The location of the My Documents folder

options index page

Provides the entry point and structure to support the Options functionality. The structure is a table that Actuate Information Console uses to format and present files and folders data. The default table includes the banner across the top of the page, the side menu on the left side of the page, and a container for page content. Page content varies depending upon the Actuate Information Console directive.

The options index page uses the banner page to provide the reporting web page banner. options.jsp uses the HTML code in <context root>\portal\activePortal\private\options\optionspage.jsp to display the options data.

Name <context root>\options.do

<context root>\portal\activePortal\private\options\options.jsp

Parameters Table 4-16 describes the parameter for the options index page. The options index page also uses the common URI parameters.

Table 4-16 Parameter for options index URI

URI parameter	Description
homeFolder	Link to My Documents

requests index page

Provides the outermost structure for the active request functionality. The requests index page displays the side menu and banner elements, and the tabbed property sheets defined by tabs. selectjobs.jsp uses the HTML code in <context root>\portal\activePortal\private\jobs\selectjobscontent.jsp to display request data.

Name <context root>\selectjobs.do

<context root>\portal\activePortal\private\jobs\selectjobs.jsp

Parameters Table 4-17 lists and describes the parameters for the requests index page. The requests index page also uses the common URI parameters.

Used by Not applicable.

Table 4-17 Parameters for request index URI

URI parameter	Description
applyFilter	Specifies whether to apply cbFail, cbSuccess, and filter to the current user session. applyFilter applies only to list pages, such as the completed jobs page.
cbFail	Specifies whether to list the failed jobs in the completed jobs page.
cbSuccess	Specifies whether to list the successful jobs in the completed jobs page.
channelName	Specifies the channel to which a job completion notice was sent. channelName applies only to the details page.
clearFilter	Clears the job name filter. clearFilter causes Actuate Information Console to retrieve job names from session cookies and to ignore cbFail and cbSuccess. clearFilter applies only to list pages, such as the completed jobs page.
filter	Specifies the job name filter. filter applies only to list pages, such as the completed jobs page.
jobID	Specifies the unique job identifier. jobID applies only to the details page.
resetFilter	Resets all filters to their default values. The default filter values are no filtering for job name, and listing all completed jobs, whether failed or successful. resetFilter applies only to list pages such as the completed jobs page.
subpage	Determines the content page. Possible values for subpage are: <ul style="list-style-type: none">■ _completed■ _detail■ _pending■ _running■ _scheduled _completed is the default content page.
userName	Specifies the name of the user who received the completed job notice. userName applies only to the detail page.

license page

Displays the License page, containing information about the Actuate Information Console version and options. Called when the user chooses the License tab on the Options page.

The default license page for Information Console is similar to Figure 4-6.



Figure 4-6 Information Console license page

Name	<code><context root>\portal\activePortal\private\options\license.jsp</code>
Parameters	The about page uses the common URI parameters.
Used by	<code>portal\activePortal\private\options\optionspage.jsp</code>

list page

Lists files in a container, such as a channel or folder. There are three types of lists:

- channels
- channel contents
- filefolders

channels list page

Lists the channels that the user subscribes to. Users can also subscribe or unsubscribe to channels from this page.

A channels list page looks like Figure 4-7. Users choose a channel name to see the contents of the channel.

Name	<code><context root>\selectchannels.do</code>
Used by	Not applicable.

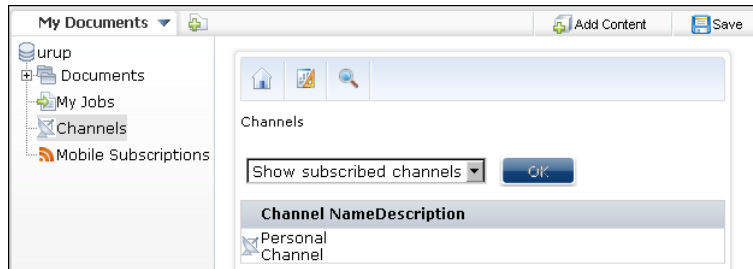


Figure 4-7 Channels list page

channel contents list page

Lists the contents of a specified channel. You cannot access this page directly, but you can edit it to change its appearance. `channelnoticelist.jsp` uses the HTML code in `<context root>\portal\activePortal\private\channels\channelnoticelistcontent.jsp` to display the contents.

A channel contents list page looks like Figure 4-8. Users choose the file or version name to view the report document and the magnifying glass for report details.

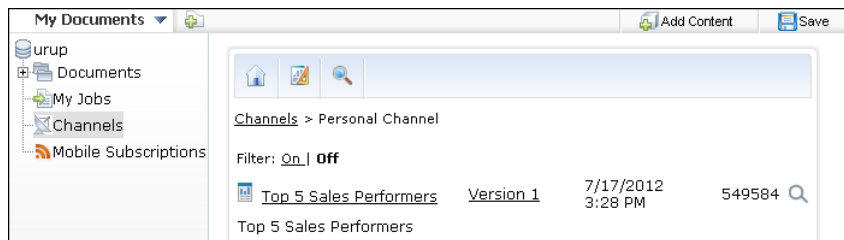


Figure 4-8 Channel contents list page

Name `<context root>\selectjobnotices.do`

Parameters Table 4-18 describes the parameter for the channel contents list page. The channel contents list page also uses the common URI parameters.

Table 4-18 Parameter for channel contents list URI

URI parameter	Description
<code>__channel</code>	The name of the channel to list

Used by Not applicable.

file and folder list page

Presents a list of objects that reside in the current working repository folder. Users request folder listings by choosing links on the reporting web page. The file and

folder list page includes a filter section where users specify criteria for viewing report documents. For example, users select check boxes to indicate whether they want to view only the last version of a report document or to see report executable files and report documents.

When users access a repository for the first time, Actuate Information Console displays their home folder, if they have one, or the top folder in the repository. All files and folders in that folder that they have permission to view appear in the Actuate Information Console listing page. Users can specify a filter to choose the types of files to view.

The following are the sources that the file and folder list page uses to obtain the values for filters and the state of check boxes:

- URI parameters. See the following parameters section.
- Session attributes. Actuate Information Console uses session cookies to store the values that a user specifies. If the user browses the Actuate Information Console application, then returns to the listing page, the list page obtains the user's values from the session cookie if cookies are enabled. If the user chooses another folder, that folder becomes the working folder, and the list page applies the same values that applied to the previous folder.

Table 4-19 lists and describes the session attribute variables.

Table 4-19 Session attribute variables

Session attribute	Description
AcFilesFoldersFilter	Contains the string specifying the files and folders viewing filter
AcFilesFoldersType Filter	Contains True if the user specified a filter, False otherwise
Name	<context root>\getfolderitems.do <context root>\portal\activePortal\private\filesfolders\filefolderlist.jsp
Parameters	Table 4-20 lists and describes the parameters for the file and folder list page. The file and folder list page also uses the common URI parameters.

Table 4-20 Parameters for file and folder list URI

URI parameter	Description
applyFilter	If True, apply filter. If False, filter not applied.
filter	The filter specifying the file and folder names to list. Filter is a string. The default is "".

(continues)

Table 4-20 Parameters for file and folder list URI

URI parameter	Description
folder	The folder for which to list the contents. Folder name is a string. If no folder is specified, List uses the last working folder known for the session if cookies are enabled. If cookies are not enabled, List uses the user's home folder as specified in the user settings.
onlyLatest	If True, show only the latest version of a file if multiple versions exist. If False, show all versions of a file if multiple versions exist. The default is False.
resetFilter	Any non-null value for resetFilter causes the filter to return to its original state. Users can reset the filter by choosing the Default button on the listing page.
showDocument	If True, show all viewable documents. If False, do not show viewable documents. The default is True.
showExecutables	If True, show all report executables. If False, do not show report executables. The default is True.
showFolders	If True, show all folders. If False, do not show folders. The default is True.

Used by Not applicable.

login banner page

Displays the Actuate Information Console web application banner. Banner elements include the company logo, system name, and help link.

Name <context root>\portal\activePortal\private\login_banner.jsp

Used by portal\activePortal\private\login.jsp

login page

Displays the Actuate Information Console login page for logging in to the Actuate Information Console web application. The login page includes the login banner page to display the Actuate Information Console application banner.

Name <context root>\login.do
<context root>\portal\activePortal\private\login.jsp

Parameters Table 4-21 lists and describes the parameters for the login page. The login page also uses the common URI parameters.

Table 4-21 Parameters for login page URI

URI parameter	Description
loginPostBack	False to display the login page and True to display the destination page instead of the login page if the login is successful.
targetPage	Specify a relative URI to which login redirects the user on successful login. The default is the file and folder list page.

Used by Not applicable.

logout page

Ends the user's Actuate Information Console session. The logout page gathers the user's session information, clears it, and returns the user to the login page.

Name <context root>\logout.do

Parameters Table 4-22 lists and describes the parameters for the logout page. The logout page also uses the common parameters.

Table 4-22 Parameters for logout page URI

URI parameter	Description
daemonURL	Contains the URI that accesses the Process Management Daemon, such as http://Server:8100.
user	The name of the user to log out. Either user or the common URI parameter authID must be specified. If authID is specified, user is ignored.

Used by Not applicable.

My dashboard page

A property sheet that supports specifying the default dashboard and resetting the layout of the default dashboard.

A My dashboard page looks like Figure 4-9.

Name <context root>\options.do

Used by iportal\activePortal\private\options\optionspage.jsp

The screenshot shows the 'My dashboard' tab with the following settings:

- Reset my dashboard to:**
 - ☐ Blank dashboard
 - ☒ System default
 - ☐ Shared dashboard:
- Default layout:**
 - ☐ One Column
 - ☒ Two Columns
 - ☐ Three Columns
 - ☐ Free Form
- Free form layout default settings:**
 - ☒ Show Grid
 - ☒ Snap to Grid
 - Grid Spacing: px

Figure 4-9 My dashboard page

notification page

A property sheet that supports specifying notification options for the current user. Notification options include whether to generate e-mail on completion of requests. A notification page looks like Figure 4-10.

The screenshot shows the 'Notification' tab with the following settings:

- For jobs that succeed:**
 - ☐ Send me an e-mail notification
 - ☒ Place a job completion notice in the Personal Channel
- For jobs that fail:**
 - ☐ Send me an e-mail notification
 - ☒ Place a job completion notice in the Personal Channel
- Note:** These settings apply for notifications that you receive.

Figure 4-10 Notification page

Name <context root>\options.do

Used by iportal\activePortal\private\options\optionspage.jsp

options page

Updates the user options and settings on the server.

An options page looks like Figure 4-11.

Name <context root>\options.do
<context root>\iportal\activePortal\private\options\options.jsp

The screenshot shows a web application interface with three tabs: 'General', 'My Dashboard', and 'Notification'. The 'General' tab is selected. Below the tabs, there are several form fields: 'Home folder:' with the value '/Home/administrator', 'E-mail address:' with an empty text box, 'Skin:' with a dropdown menu showing 'Tree View Skin', and 'View:' with a dropdown menu showing 'Categories'. There are two checkboxes: 'Enable filters:' with the label 'Display Filter for Channels, Documents and Jobs', and 'Document viewing:' with the label 'Open in new browser window'. Both are currently unchecked. Below these is a section titled 'Update password:' with three text boxes for 'Old password:', 'New password:', and 'Re-enter new password:'. At the bottom left of the form is a blue button labeled 'Save Options'.

Figure 4-11 Options page

Parameters Table 4-23 lists and describes the parameters for the options page. The options page also uses the common URI parameters.

Table 4-23 Parameters for options URI

URI parameter	Description
channelIcons	Specifies whether or not to display channel icons.
channels	Contains the string list of channels to which the user subscribes.
confirmKey	Contains the user's password.
docChanFilters	Specifies filters for viewing documents or channels.
email	Contains the user's e-mail address.
failComp	Indicates whether to generate completed request notifications for failed jobs. Enable to generate notifications for failed requests, Disable otherwise.
failEmail	Indicates whether to generate e-mail for failed requests. Set the value to "on" to enable or "off" to disable.

(continues)

Table 4-23 Parameters for options URI

URI parameter	Description
newKey	Contains the user's new password.
newLocale	Contains the user's new locale.
newTimeZone	Contains the user's new time zone.
oldKey	Contains the user's old password.
redirect	Specifies the page to go to when user options update is complete.
requestFilters	Indicates whether to use filters for the Request page. Enable to use filters, Disable otherwise.
succComp	Indicates whether to generate completed request notifications for successful requests. Enable to generate notifications for failed requests, Disable otherwise.
succEmail	Indicates whether to generate e-mail for successful requests. Set the value to "on" to enable or "off" to disable.
userName	Contains the current user's name.
viewNewBrowser	Indicates whether to view documents in the current browser window or in a new browser window. Set the value to "on" to view documents in a new browser window or "off" to disable.

Used by Not applicable.

output page

Specifies report executable output data, such as the report headline and output file name. The output page appears only for scheduled report job and Run and Save report job submissions. Users access Output by choosing Save As.

An output page looks like Figure 4-12.

Name <context root>\portal\activePortal\private\newrequest\output.jsp

Parameters Table 4-24 lists and describes the parameters for the output page. The output page also uses the common URI parameters.

Table 4-24 Parameters for output URI

URI parameter	Description
headline	Specifies the headline for the report.
ifExists	Specifies the file replacement policy. Values are Create and Replace. If ifExists is Create, Information Console creates a new version. If ifExists is Replace, Information Console replaces the existing version.

Table 4-24 Parameters for output URI

URI parameter	Description
outputFolderType	Specifies the report output’s folder type. Values are personal and absolute. If outputFolderType is personal, the output is placed in the user’s personal folder. If outputFolderType is absolute, the user specifies the full path name for the output by either typing the path or using the Browse button.
outputName	Specifies the name of the output file.
versionName	Specifies the version name.

Figure 4-12 Output page

Used by `iportal\activePortal\private\newrequest\newrequestpage.jsp`

page not found page

Displays an error message when Information Console cannot find the page that a user specifies.

Name `<context root>\iportal\activePortal\errors\pagenotfound.jsp`

Used by Not applicable.

parameters page

Displays report job parameters. Parameters include the headline, output file name, and report executable file name. Users access the parameters list by choosing Parameters.

The parameters page looks like Figure 4-13.



Figure 4-13 Parameters page

Name	<context root>\portal\activePortal\private\newrequest\parameters.jsp
Used by	portal\activePortal\private\newrequest\newrequestpage.jsp

pending page

Lists all jobs that are currently awaiting execution.

Name	<context root>\portal\activePortal\private\jobs\pendingjob.jsp
Parameters	The pending page uses the common URI parameters.
Used by	portal\activePortal\private\jobs\selectjobscontent.jsp

ping page

The ping page tests whether a specific component of the reporting environment is operational, and optionally retrieves other diagnostic information about the component. You can test the following components of the reporting environment:

- Information Console itself
- The Encyclopedia service
- The Factory service
- The Integration service
- The Message Distribution service (MDS)

- The View service
- An Actuate open server driver

If a component is not operational, Actuate BIRT iHub returns an error message. If a component is operational, the response depends on the ping page parameters. For example, you can request a simple time stamp that shows the time elapsed between the time that a component receives the request and the time that it returns a reply, as shown with the following URI:

`http://seamore:8700/iportal/ping.do?destination=EE&mode=trace`
generates the following response:

```
18:03:23.100: MDS(seamore) received Ping message
18:03:23.100: MDS(seamore) forwarding Ping request to node seamore
18:03:23.100: EncycEngine(seamore) received Ping message
EncycEngine(seamore): Echoing 0 bytes of payload data
18:03:23.100: EncycEngine(seamore) replying to Ping message.
    Elapsed=0 ms
18:03:23.100: MDS(seamore) received Ping reply from node seamore.
    Roundtrip= 0 ms
18:03:23.100: MDS(seamore) replying to Ping message. Elapsed=0 ms
```

You also can request more detailed information. A ping request to the MDS has no security restrictions. For all other components, the request is subject to Encyclopedia volume authentication. The user must be an Encyclopedia volume administrator or a user with the Operator security role.

Name	<context root>\ping.do
Parameters	Table 4-25 lists and describes the parameters for the ping page. The ping page also uses the common URI parameters.

Table 4-25 Parameters for ping URI

URI parameter	Description
action	<p>Specifies the action to take at the destination. Valid values are:</p> <ul style="list-style-type: none">■ Echo—Echoes data specified by the payloadSize parameter. Echo is the default action.■ ReadFile—Opens a specified Encyclopedia volume file, reads its content, and closes the file. Destination must be EE, FS, or VS.■ WriteFile—Creates a temporary file in a partition, writes a specified number of bytes, closes the file, and deletes it. Destination must be EE or FS.■ Connect—Connects to a data source.■ If you do not specify a value, the destination component responds to the request without taking any other actions.

(continues)

Table 4-25 Parameters for ping URI (continued)

URI parameter	Description
destination	<p>The reporting environment component to test. Valid values are:</p> <ul style="list-style-type: none"> ■ AP (Information Console) ■ MDS (Message Distribution service) ■ EE (Encyclopedia Engine) ■ FS (Factory service) ■ VS (View service) ■ AIS (Actuate Integration service) <p>AIS only supports the Echo action.</p> <p>Except when AP is specified as destination, Actuate Information Console sends a ping request to the Actuate BIRT iHub and passes on the destination as the ping request's destination parameter.</p> <p>The default value is AP.</p>
filename	<p>When action=ReadFile, this parameter is required to indicate the Encyclopedia volume file to read. If you ping an open server driver, filename specifies the executable file to prepare for execution.</p>
mode	<p>Specifies the level of detail in the ping response. Valid values are:</p> <ul style="list-style-type: none"> ■ Concise—Returns the elapsed time between a component's receipt of the request and the time the component sends a reply. ■ Normal—Returns the names of components in the test path and the time stamps of the request entering and leaving each component. This is the default mode. ■ Trace—Returns a time stamp at times when the request enters and leaves major subcomponents of the component being tested. For example, a request to a node running the Encyclopedia service can provide a time stamp for times when the request enters and leaves the process queue. <p>A ping request at the trace level also can return diagnostic information other than timing. For example, a request to test writing a temporary file to a partition can return the amount of free disk space on the partition.</p>
partitionName	<p>Specifies the name of the Encyclopedia partition on which to create the temporary file. Used only if the value of action is WriteFile.</p>
payloadSize	<p>Length of payload string in number of characters that Actuate Information Console should generate. Used only if the value of action is Echo.</p>
processID	<p>Specifies the process ID of the Factory or View service to test. Used with the server parameter.</p>

Table 4-25 Parameters for ping URI (continued)

URI parameter	Description
server	Specifies which instance of a Factory service or View service to test. Works with the processID parameter. To test all available instance of the Factory or View service, use an asterisk (*). If you do not specify server, the Actuate BIRT iHub load balancing mechanism allocates an available instance of the requested service to respond to the ping request.

Used by Not applicable.

privileges page

Assigns privileges to a file or folder. Filefoldersprivilege.do uses the HTML code in <context root>\portal\activePortal\private\filefolders\filefolderlist.jsp to set the privileges. The following URI displays the privilege page for the hotgraph report executable in the Training folder:

`\portal\filefoldersprivilege.do?name=\Training\hotgraph.rptdesign`

Name <context root>filefoldersprivilege.do

Parameters Table 4-26 lists and describes the parameter for the privileges page. The privileges page also uses the common URI parameters.

Table 4-26 Parameters for privileges URI

URI parameter	Description
name	File or folder name to set privileges for

Used by `portal\activePortal\private\common\popupmenu.jsp`
`portal\activePortal\private\filefolders\filedetailcontent.jsp`

running page

Lists all jobs that were executing when the running page was last refreshed. The list is not live. To view the current list, the user must refresh the browser. Users access the running jobs list by choosing Running.

The running page looks like Figure 4-14.

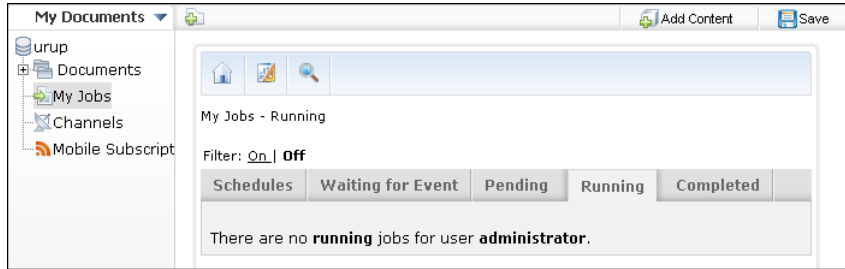


Figure 4-14 Running page

Parameters The running page uses the common URI parameters.

Name <context root>\portal\activePortal\private\jobs\runningjob.jsp

Used by iportal\activePortal\private\jobs\selectjobscontent.jsp

schedule page

Supports specifying report executable file run schedules. The schedule page applies only to scheduled report job requests.

Schedule properties include data and time for running the request, recurring schedules to run a report job on a regular basis, or whether to run the report job immediately.

The Information Console schedule page is similar to Figure 4-15.

<context root>\js\calendar.js provides calendar functionality for Information Console. Note that date and time field lengths are hard-coded in the schedule page.

Figure 4-15 Information Console schedule

Name <context root>\portal\activePortal\private\newrequest\schedule.jsp

Parameters Table 4-27 lists and describes the parameters for the schedule page. The schedule page also uses the common URI parameters.

Table 4-27 Parameters for schedule URI

URI parameter	Description
jobName	The name of the request being submitted.
onceDate	If scheduleType is once, specify the date on which to run the report job.
onceTime	If scheduleType is once, specify the time at which to run the report job.
recurringDay	The day on which to run the request on a regular basis. Values are the day of the week, EVERYDAY, FIRST_DAY_OF_THE_MONTH, LAST_DAY_OF_THE_MONTH.
recurringTime	If scheduleType is recurring, specify the time at which to run the report job.
scheduleType	Specify the schedule type. Values are immediate, once, and recurring.

Used by portal\activePortal\private\newrequest\newrequestpage.jsp

scheduled job page

Lists all jobs that activate at a specified date and time but are not yet active.

Name <context root>\portal\activePortal\private\jobs\scheduledjob.jsp

Parameters The scheduled job page uses the common URI parameters.

Used by Not applicable.

search folders page

Recursively searches from the current folder for files and folders whose names match the search string.

Name <context root>\searchfiles.do

Parameters Table 4-28 lists and describes the parameters for the search folders page. The search folders page also uses the common URI parameters.

Table 4-28 Parameters for search folders URI

URI parameter	Description
folder	Folder name to start the search from. The default is the current location, as shown in the breadcrumb.
searchFilter	The name to search for. Expressions and wildcards are allowed. For more information about search expressions, see <i>Using Information Console</i> .

For example, the following Information Console URL searches the current folder and all subfolders for files or folders whose names begin with the string Cust:

`http://localhost:8700/iportal/searchfiles.do?searchFilter=Cust*`

A search results page looks like Figure 4-16.

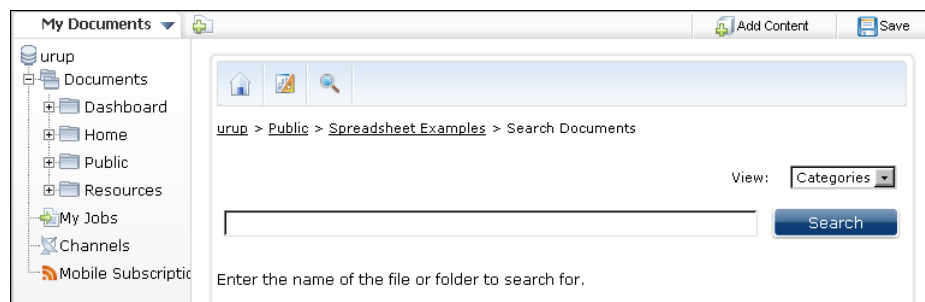


Figure 4-16 Search results

Used by Not applicable.

submit job page

Submits a scheduled report job for a report executable to the server. There is no user interface to the submit job page. `submitjobstatus.jsp` uses the HTML code in `<context root>\iportal\activePortal\private\newrequest\submitjobstatuspage.jsp` to display the new request information.

For reports that accept run-time parameters, set the parameter in the URL by adding an ampersand (&), the parameter name, and an equal (=) sign, followed by the parameter value in quotation marks.

Name `<context root>\submitjob.do`

`<context root>\iportal\activePortal\private\newrequest\submitjobstatus.jsp`

Parameters Table 4-29 lists and describes the parameters for the submit job page. The submit job page also uses the common URI parameters. All other parameters are passed

to the report executable as report parameters. Report parameters are case-sensitive. Specify them exactly as defined in the report design.

Table 4-29 Parameters for submit job URI

URI parameter	Description
__accessToGrant	<p>Grants read or secure read privileges to those roles that have permission to view the report document. For users to view only the parts of the document matching an access control list (ACL), grant Secure Read access. Otherwise, grant Read access to enable users to view the whole document.</p> <p>This parameter requires the __channels, __exclude, and invokeSubmit=true parameters, even if you use no value for them.</p> <p>Use the __exclude parameter with this parameter to exclude specific users from getting the privilege.</p> <p>Use the __channels parameter to grant read privileges to channels and notify them.</p>
__ageDays	<p>Used with __ageHours to determine how long output objects exist before they are deleted. Use only if __archivePolicy is set to age. __ageDays can be any positive number.</p>
__ageHours	<p>Use with __ageDays to determine how long output objects exist before they are deleted. Use only if __archivePolicy is set to age. __ageHours can be any positive number.</p>
__archiveBeforeDelete	<p>Indicates whether to archive the output objects of the request before deleting them, according to __archivePolicy's setting. Set this parameter to True to archive objects before deleting them. The default value is False.</p> <p>This parameter has no effect if __archivePolicy is set to folder.</p>
__archivePolicy	<p>The archive policy to implement for the objects created as output for the request. Values are folder, age, and date. Set this parameter to folder to use the archive policy already set for the folders to which the output is distributed, to age to delete objects older than a specific time period, or to date to delete objects on a specific date.</p>
__dateToDelete	<p>The date on which to delete the output objects of the current request. Use only if __archivePolicy is set to date. __dateToDelete must be a date in a locale-specific format. The default format is mm/dd/yyyy.</p>
__executableName	<p>The name of the executable file for this request.</p>
folderType	<p>Specifies the destination folder type for the report. Absolute indicates the repository root folder, /. Personal indicates the current user's home folder. Default is Personal.</p>

(continues)

Table 4-29 Parameters for submit job URI (continued)

URI parameter	Description
__headline	A descriptive tag line for a report document. Appears on the Channel Contents page. Use the character string %20 to represent spaces in the headline string.
__ifExists	Indicates whether to overwrite an existing or create a new file, up to an optional limit. Values are: <ul style="list-style-type: none"> ■ create—creates a new output file. ■ create[n]—creates a new output file up to n versions. For example, to create no more than seven versions, use create7. ■ replace—overwrite any existing output.
invokeSubmit	Controls whether the browser is redirected to the parameter screen or whether the report job is scheduled immediately. If True, the report job is scheduled without displaying the parameters. If False, the parameters are displayed. False is the default.
__jobName	The name for the job to submit.
notificationSupported	Specifies whether to notify users who have notification disabled. True sends notification and disregards user preferences. Default value is False.
notify	Activates e-mail notification for the job.
__onceDate	Required for once schedules. Specify the date on which to run the report job, for report jobs with __scheduleType of once. Must be in the appropriate format for your locale, such as mm/dd/yyyy for the U.S. locale. The current date is the default.
__onceTime	Required for once schedules. Specify the time at which to run the report job, for report jobs with __scheduleType of once. Must be in the appropriate format for your locale, such as “hh:mm a” for the U.S. locale. The current time is the default.
__outputName	Specifies a name for the report output document.
outputName	Specifies a name for the report output document for the e-mail notification.
outputFormat	Optional parameter that appends a file extension to the outputName. Do not use a period in the value of this parameter, a period is inserted automatically before the file extension.
postback	Forces the browser not to display parameters. Set to False to display parameters. Do not set postback to True with invokeSubmit also set to True.

Table 4-29 Parameters for submit job URI (continued)

URI parameter	Description
<code>--priority</code>	Specifies the job submission priority. Values are a number from 1 to 1000, High (800), Medium (500), and Low (200). Do not use with <code>--priorityValue</code> .
<code>--priorityValue</code>	Specifies a number corresponding to the job submission priority. Do not use with <code>--priority</code> .
<code>--progressive</code>	Indicate whether to display the report document after it generates. If False, the report document displays after it generates. If True, the report document displays progressively, as it generates. Applies only to run report jobs.
<code>--recurringDay</code>	Specifies the scheduled recurring day on which to run the report job. Applies only to scheduled report jobs.
<code>--recurringTime</code>	Required for recurring schedules. Specify the time at which to run the report job. Set only if report jobs <code>--scheduleType</code> is recurring. Must be in the appropriate format for your locale, such as hh:mm:ss for the U.S. (enu) locale.
<code>--redirect</code>	Specifies a relative or absolute URI to go to after <code>do_executereport</code> submits the report job. The default is <code>Submittedjob_Status</code> .
<code>--schedulePeriod</code>	Required for recurring schedules. Specify how often to run the report job, and on which days. Choose a day of the week. <code>--schedulePeriod</code> values are Every Day, Weekdays, Mondays, Tuesdays, Wednesdays, Thursdays, Fridays, Saturdays, Sundays, First Day of the Month, Last Day of the Month. All values are case-sensitive. Every Day or Weekdays. Set only if <code>--scheduleType</code> is recurring.
<code>--scheduleType</code>	Specify the type of schedule: immediate, once, or recurring. Immediate is the default.
<code>--serverURL</code>	Contains the URI that accesses the JSP engine, such as <code>http://Services:8700</code> .
<code>--timeToDelete</code>	Specifies a time at which to delete an archived report document. Applies only to scheduled report jobs.
<code>--versionName</code>	Contains a string value for the new version name of the job's report document output. The value can include a date/time expression enclosed in braces, {}, to ensure a unique version name.
<code>--volume</code>	Contains a string value specifying the volume for the job's report document output.

The submit job page also accepts dynamic filter parameters for BIRT Reports in the URL, but the value of the parameter must form a complete expression, such as `&Territory=([Territory] = "EMEA")`.

For example, the following URL schedules the Sales By Territory.rptdesign report to run once on the September 16, 2010 with the Territory run-time parameter set to Japan:

```
http://localhost:8700/iportal/  
submitjob.do?__requesttype=scheduled&__executableName=%2fPublic  
%2fBIRT%20and%20BIRT%20Report%20Studio%20Examples%2fSales%20by%  
20Territory%20erptdesign%3b1&userid=administrator&__scheduleType  
=once&__onceDate=09/16/2010&__onceTime=1:55  
pm&Territory="Japan"&invokeSubmit=True
```

Used by iportal\activePortal\private\filesfolders\filefolderlistcontent.jsp
 iportal\activePortal\private\newrequest\newrequestpage.jsp

Actuate BIRT Viewer URIs reference

To view and interact with Actuate BIRT reports, you use the Actuate BIRT Viewer servlet. All BIRT Viewer options and varieties use the same URI. For detailed information about the BIRT Viewer servlet URI, see *Working with Actuate BIRT Viewers*.

Actuate Information Console JavaScript

This chapter contains the following topics:

- Actuate Information Console JavaScript overview
- Actuate Information Console JavaScript reference

Actuate Information Console JavaScript overview

This section describes the Actuate Information Console JavaScript files. Actuate Information Console JavaScript files provide functionality and dynamic content to Actuate Information Console web applications. Actuate Information Console JavaScript files reside in <context root>\portal\js.

Actuate Information Console JavaScript reference

Table 5-1 lists and describes the Actuate Information Console JavaScript files.

Table 5-1 Information Console JavaScript files

Name	Description
allscripts.js	Defines global variables, resources, and common methods such as deleteFile and viewActiveRequests
array.js	Contains functionality for handling arrays and array elements
browsertype.js	Determines the web browser in use and provides functionality appropriate to the browser, such as opening a file in a new window and capturing a keystroke event
calendarlayer.js	Provides calendar functionality for Information Console
converter.js	Provides character encoding
cookie.js	Provides cookie functionality, including reading, writing, and clearing browser cookies
drift.js	Adjusts layers and window display for Information Console
encoder.js	Contains the encode and unencode methods
help.js	Provides context-sensitive help functionality for Information Console
htmlselect.js	Provides methods for manipulating option controls
layer.js	Provides layer functionality, such as createLayer, deleteLayer, getWidth, showLayer
popupmenu.js	Defines the methods for manipulating pop-up menus
report.js	Provides the JavaScript components for report viewing

Table 5-1 Information Console JavaScript files

Name	Description
resize.js	Provides the JavaScript component for resizing a page for Information Console
saveas.js	Provides the JavaScript component for saving a file as another file or object
skincustomization.js	Provides the JavaScript components for maintaining Actuate Information Console skins
strutscommon.js	Provides JavaScript components for using the Struts framework with Information Console
viewnav.js	Displays the Actuate Information Console toolbar

Actuate Information Console servlets

This chapter contains the following topics:

- Information Console Java servlets overview
- Information Console Java servlets reference

Information Console Java servlets overview

Java servlets extend web server functionality. Information Console uses Java servlets to manage binary content and to perform tasks such as uploading and downloading binary files. Actuate provides an abstract framework of servlets that provide common functionality to Information Console and Management Console. You cannot modify the Actuate Java servlets.

About the base servlet

All Actuate servlets derive from the base servlet:

```
com.actuate.reportcast.servlets.AcServlet
```

The base servlet has no URI parameters. It provides Actuate servlets with the functionality for performing the following tasks:

- Parse and validate parameters specified in Information Console URI directives.
- Create XML API structures based on Actuate Information Console requests.
- Submit XML streams to the Actuate SOAP API.
- Handle responses from the Actuate SOAP API, including detecting errors.
- Store constant session information, such as the name space and SOAP endpoint.
- Read from and write to cookies.
- Stream report data or errors to the web browser.

Invoking a servlet

Invoke servlets using the following syntax:

```
http://<application server>:<port>/<context root>/servlet  
/<servlet alias>
```

- application server is the name of the machine hosting the application server.
- port is the port on which the application server listens for requests.
- context root is the Information Console context root.
- servlet is a keyword indicating that a servlet follows.
- servlet alias is the name to which the servlet is mapped in the Information Console installation's web.xml file. A typical location for web.xml is C:\Program Files (x86)\Actuate\iPortal2\iportal\WEB-INF\web.xml.

Servlet names are case-sensitive. Do not modify the servlets, their names, or their mapping in web.xml.

Information Console Java servlets reference

Table 6-1 lists and describes the Information Console Java servlets.

Table 6-1 Actuate Information Console servlets

Information Console servlet	Description
DownloadFile servlet	Downloads a file from the Encyclopedia volume
Interactive Viewer servlet	Displays a BIRT report document

This section provides the detailed reference for Information Console servlets.

DownloadFile servlet

Downloads a file from the Encyclopedia volume.

Name com.actuate.reportcast.servlets.FileDownloadServlet

Invoke the DownloadFile servlet as:

http://<web server>:<port>/<context root>/servlet/DownloadFile

URI parameters Table 6-2 lists and describes the URI parameters for the DownloadFile servlet.

Table 6-2 Parameters for DownloadFile URI

URI parameter	Description
fileId	The unique identifier of an object, usually retrieved with the selectFilesFolders JSP tag.
name	The name of the object to download.
version	If name is specified, the version number of the object to view. If version is not specified, the latest version is retrieved.

Interactive Viewer servlet

Displays an Actuate BIRT report document with tools to affect the document and design files. The viewer has two modes, standard and interactive.

The Standard Viewer displays the report with toolbar options to save, print, show the TOC, and launch interactive mode, as shown in Figure 6-1.



Figure 6-1 Standard Viewer

The Interactive Viewer displays the report with toolbar options to navigate the report and provides context menus to edit and format report elements, as shown in Figure 6-2.



Figure 6-2 Interactive Viewer

Name com.actuate.iv.servlet.IVServlet

Invoke the Interactive Viewer servlet as:

http://<web server>:<port>/<context root>/iv

URI parameters Table 6-3 lists and describes the URI parameters for the Interactive Viewer servlet.

Table 6-3 Parameters for IV URI

URI parameter	Description
__bookmark	Name of the element of a report to display instead of the whole report file
__floatingfooter	Boolean value to add a margin under the footer
__format	A format for the displayed report: <ul style="list-style-type: none"> ■ pdf: Adobe PDF ■ xls: MS Excel ■ doc: MS Word ■ ppt: MS PowerPoint ■ ps: PostScript ■ html: HTML ■ flashchartsxml, flashgadgetsxml: used to display a fusion chart ■ reportlet: used together with __bookmark to show a particular part/element of the report

Table 6-3 Parameters for IV URI

URI parameter	Description
__from_page_range	The page range of a report to display
__from_page_style	The page style to use for a report in pdf or ps formats: <ul style="list-style-type: none"> ■ auto: The page size and content size remains the same. ■ actuateSize: Change the page size to fit the content. ■ fitToWholePage: Change the content size to fit the page size. Used with the __format parameter
__imageid	Name of the report file to display
__instanceid	Name of the report file to display
__launchiv	Boolean value that enables interactivity
__locale	Code for a locale
__page	A number for a page to render
__report	Name of the report file to display
__rtl	Name of the report file to display
repositoryType	The name of the object to download
serverURL	Contains the URL that accesses iHub, such as http://ESL02835:8000
userid	The user's identifier, required to log in to Actuate BIRT iHub
volume	Contains a string value specifying the volume for this report

Actuate Information Console custom tags

This chapter contains the following topics:

- Information Console custom tag overview
- Information Console custom tags quick reference
- Information Console custom tags reference

Information Console custom tag overview

This chapter provides reference information about Information Console tag libraries and their custom tags. Custom tags are JSP language elements that you define to encapsulate frequent tasks. A tag library defines a set of related custom tags and contains the objects that implement the tags.

The Information Console tag libraries reside in <context root>\WEB-INF. The tag libraries define the XML tags and attributes that the Information Console pages use. Examine individual pages to determine the tag libraries that they use. For example, viewdefault.jsp uses the internationalization, common, and users tag libraries, as shown in the following example:

```
<%-- DECLARE ANY RESOURCE BUNDLES USED IN THIS PAGE --%>
<%@ taglib uri="/i18n" prefix="i18n"%>
<%@ taglib uri="/common" prefix="common" %>
<%@ taglib uri="/users" prefix="users" %>
```

You declare that a page uses tags by including the taglib directive in the page before you use any custom tag. The uri attribute refers to a URI (Uniform Resource Identifier) that uniquely identifies the tag library descriptor (TLD). A TLD file is an XML document that describes a tag library. The prefix attribute defines the prefix that distinguishes tags defined by a given tag library from those provided by other tag libraries. The prefix can differ for each use of the taglib statement, but every prefix must be unique within a page.

Information Console custom tag names are case-sensitive.

Information Console custom tags quick reference

This section provides two quick reference lists related to Information Console custom tags:

- Information Console custom tag libraries
- Information Console custom tags

Information Console custom tag libraries

Table 7-1 lists and describes the Information Console custom tag libraries.

Table 7-1 Actuate Information Console tag libraries

Tag library	Description
actabpanel	Provides tags for creating tabbed pages
common	Provides tags storing and iterating through data

Table 7-1 Actuate Information Console tag libraries

Tag library	Description
filesfolders	Tags for managing files and folders
i18n	Provides tags for internationalization
login	Provides tags for login operation

Information Console custom tags

Information Console uses Jakarta Struts custom tags and Actuate custom tags. Actuate recommends that customized Information Console web applications use Jakarta Struts custom tags and only the Actuate custom tags shown in Table 7-2.

Table 7-2 Actuate Information Console custom tags

Tag library	Tag	Description
actabpanel	content	Specifies the page to display when the user or URL selects the associated tab
actabpanel	tab	Specifies the label on a page's tab and its key
actabpanel	tabBegin	Specifies any HTML or JSP code to apply before defining any tabs
actabpanel	tabEnd	Specifies any HTML or JSP code to apply after defining all tabs
actabpanel	tabMiddle	Specifies any HTML or JSP code to apply to each unselected tab
actabpanel	tabMiddleSelected	Specifies any HTML or JSP code to apply to the selected tab
actabpanel	tabPanel	Contains all tags defining page tabs
actabpanel	tabSeparator	Specifies any HTML or JSP code to apply between each adjacent pair of tabs
common	tab	Holds a single string of data
common	tab	Holds an array of strings
filesfolders	copyFileFolder	Copies files and folders
i18n	bundle	Wraps org.apache.taglibs.i18n.BundleTag
i18n	formatDate	Formats a Date value using a locale
i18n	message	Allows the usage of a resource bundle to internationalize content
login	login	Performs the login operation

Information Console custom tags reference

This section provides the detailed reference for Information Console custom tags.

bundle

Establishes the ResourceBundle to use for other i18n tags in the JSP. It also determines the most appropriate locale to use based on browser settings if a locale is not provided. It overrides the doEndTag() method and sets the ChangeResponseLocale feature to False. This tag must be placed in a JSP before any other i18n tags. This tag wraps the org.apache.taglibs.i18n.BundleTag.

Library i18n

Tag class com.actuate.reportcast.tags.common.BundleTag

Attributes Table 7-3 lists and describes the attributes for bundle.

Table 7-3 Attributes for bundle

Attribute	Required	Description
baseName	Yes	Used along with the locale to locate the desired ResourceBundle
id	No	Variable ID for use with standard jsp:getProperty tag and as an attribute to other tags in this tag library
locale	No	Current user's locale, such as en_US, from <context root>\WEB-INF\localemap.xml
localeAttribute	No	Name of an attribute whose value is the user's preferred locale

Variables Table 7-4 describes the variable for bundle.

Table 7-4 Variable for bundle

Variable	Description
id	Allows other tags or scriptlets to access the ResourceBundle defined by this tag. This is useful for allowing multiple bundle declarations per page or for creating localization debug pages by listing all key and value pairs in a bundle.

Example The following line defines a bundle using browser preference to determine locale:

```
<i18n:bundle baseName="com.mycorp.taglibs.i18n.test"/>
```

The next example defines a bundle using browser preference to determine locale, and declaring the scripting variable bundle:

```
<i18n:bundle baseName="com.mycorp.taglibs.i18n.test" id="bundle"/>
```

The next example defines a bundle using a scriptlet variable to specify the locale:

```
<i18n:bundle baseName="com.mycorp.taglibs.i18n.test"
  locale="<%= localeVar %>"/>
```

Used in <context root>\errors\error.jsp
<context root>\errors\pagenotfound.jsp

content

Specifies the content of a page for a tab. Include HTML or JSP code in the body of the content tag or use the page attribute to include another JSP file as the content.

Library actabpanel

Tag class com.actuate.activeportal.tags.tabpanel.Content

Attributes Table 7-5 describes the attribute for content.

Table 7-5 Attribute for content

Attribute	Required	Description
page	No	Specifies a file containing the JSP code to use as the content of the page associated with the current tab

If you do not include a page attribute, any HTML or JSP code in the tag's body becomes the definition of the page.

Used in <context root>\private\jobs\selectjobscontent.jsp
<context root>\private\newrequest\newrequestpage.jsp
<context root>\private\options\optionspage.jsp

Examples The following example uses the page attribute to specify using the code in saveas.jsp as the Save As tab's page content:

```
<ui:tab><bean:message key="TAB_SAVE_AS"/></ui:tab>
<ui:content page="saveas.jsp"/>
```

The following example uses the tag's body to specify the About tab's content page as the result of the JSP include directive:

```
<ui:tab key="about" unselected="class=\"lnkTab\"">
  <bean:message key="TAB_ABOUT"/>
  <ui:content><%@ include file="about.jsp" %></ui:content>
</ui:tab>
```

copyFileFolder

Copies files and folders from one location to another.

Library filesfolders

Tag class com.actuate.reportcast.tags.filesfolders.CopyFileFolderTag

Attributes Table 7-6 lists and describes the attributes for copyFileFolder.

Table 7-6 Attributes for copyFileFolder

Attribute	Required	Description
appendFileName	No	Boolean value. If True, add newName to the targetPath. Default is True.
authID	Yes	Unique authentication ID assigned to the user after successful login.
createNewVersion	No	Boolean value. If True, create a new version of the file or folder. Default is True.
latestVersionOnly	No	Boolean value. If True, only the latest version is to be copied. Default is True.
locale	No	Current user's locale, such as en_US, from <Context root>\WEB-INF\localemap.xml.
maxVersions	No	Number of versions to copy.
newName	No	New name for copied item.
serverURL	Yes	URL that accesses the BIRT iHub, such as http://Services:8000.
targetPath	Yes	Directory path to which to copy.
timeZone	No	Current user's time zone from <Context root>\Web-inf\TimeZone.xml.
volume	Yes	BIRT iHub volume to copy from.
workingFolderID	No	Unique ID for the source folder.
workingFolderName	No	Name of the source folder.

formatDate

Formats a Date value using a locale. A style or a pattern such as 'YYYY MMM ddd' is specified. If the value is null then the default text is used. If no locale is

specified then the parent locale tag is used. If no parent locale tag exists then the locale is taken from the current request. If still no locale is found then the current JVM locale is used.

- Library** i18n
- Tag class** org.apache.taglibs.i18n.FormatDateTag
- Attributes** Table 7-7 lists and describes the attributes for formatDate.

Table 7-7 Attributes for formatDate

Attribute	Required	Description
defaultText	No	Default value.
locale	No	Current user’s locale, such as en_US, as in <context root>\Web-inf\localemap.xml.
pattern	No	Date formatting string. Do not use with style.
style	No	Short, medium, long, or full. Do not use with pattern.
value	No	Date value.

- Used in** <context root>\private\channels\channelnoticelistcontent.jsp
<context root>\private\filesfolders\filedetailcontent.jsp
<context root>\private\jobs\completedjob.jsp
<context root>\private\jobs\getjobdetailscontent.jsp
<context root>\private\jobs\pendingjob.jsp
<context root>\private\jobs\runningjob.jsp
<context root>\private\jobs\scheduledjob.jsp

login

Establishes the connection to the Actuate BIRT iHub. On successful connection, Actuate BIRT iHub returns the authentication ID used during the user’s session to validate credentials and check access permissions. Actuate BIRT iHub returns the home folder and start folder as well.

- Library** login
- Tag class** com.actuate.reportcast.tags.common.LoginTag
- Attributes** Table 7-8 lists and describes the attributes for login.

Table 7-8 Attributes for login

Attribute	Required	Description
disableBasic Authentication	No	Boolean value. If True, disable basic authentication.
force	No	Boolean value. True to force a login, False to display the login page. The default is False. For example, when switching between Encyclopedia volumes and using APSE, set force=true to force the Information Console Login module to call APSE to perform the login operation. This prevents the login page from appearing unnecessarily.
id	Yes	Unique identifier for the object.
locale	No	The locale to display.
password	No	The user's password.
serverURL	No	The host and port of the Actuate BIRT iHub to which the user wants to connect.
timeZone	No	The time zone to display.
userID	Yes	The user's identifier, required to log in to the Actuate BIRT iHub.
volume	No	The name of the volume to which the user wants to connect. If volume is not specified, the login tag checks whether the variable volume_default is set in the web.xml file.

Variables Table 7-9 lists and describes the variables for login.

Table 7-9 Variables for login

Variable	Description
homeFolder	The user's home folder
startFolder	The folder that the user sees upon successful login
authID	The authentication ID returned by Actuate BIRT iHub upon successful login

Used in <context root>\authenticate.jsp

Example The following example logs in the user jaguilar to the volume sales on the Actuate BIRT iHub marcom with the password secret:

```
<actu:login clusterURL="http://marcom:8900/" userID="jaguilar"
  password="secret" volume="sales" />
```

message

Implements a body tag allowing the usage of a resource bundle to internationalize content in a web page. The key attribute is required, and is used to look up content in the resource bundle. The args attribute is optional, and if present, provides items to pass to a MessageFormat. The bundle tag must be used first in order to ensure that the proper bundle is loaded.

Library i18n

Tag class org.apache.taglibs.i18n.MessageTag

Attributes Table 7-10 lists and describes the attributes for message.

Table 7-10 Attributes for message

Attribute	Required	Description
args	No	An array of arguments for use with java.text.MessageFormat when formatting the display text
bundle	No	Object reference to the ResourceBundle in which the key can be found
bundleRef	No	Name of an attribute that contains a resource bundle
key	Yes	Key to use when retrieving the display message format from the ResourceBundle

Used in <context root>\errors\pagenotfound.jsp

Variables Table 7-11 describes the variable for message.

Table 7-11 Variable for message

Variable	Description
id	id allows other tags or scriptlets to access the String created by this tag. If id is specified the String is not printed by this tag, just stored into the id.

Example The following example displays a plain message using the default (first defined) bundle:

```
<i18n:message key="column1.header"/>
```

The next example displays a plain message using a specified bundle. In this example the default bundle is bundle1 because it is defined first:

```
<i18n:bundle baseName="com.mycorp.taglibs.i18n.i18n-test"
  id="bundle1"/> <!-- the default -->

<i18n:bundle baseName="com.mycorp.taglibs.i18n.i18n-test2"
  id="bundle2"/> <!-- the alternate -->

<i18n:message key="column1.header" bundle="<%= bundle2 %>" />
```

tab

Defines the label and key for a tab in a tab panel. URIs specifying the key cause selection of the tab and display of the page associated with the tab.

- Library** actabpanel
- Tag class** com.actuate.activeportal.tags.tabpanel.Tab
- Attributes** Table 7-12 lists and describes the attributes for tab.

Table 7-12 Attributes for tab

Attribute	Required	Description
key	No	Specifies the identification key for this tab. If not set, the default is 0, 1, 2, and so on. Use with the selectedTab attribute of the tabPanel tag.
selected	No	Specifies the label on the tab while the tab is selected.
unselected	No	Specifies the label on the tab while the tab is not selected.

Used in <context root>\private\common\errors\error.jsp
<context root>\private\common\sidebar.jsp
<context root>\private\jobs\selectjobscontent.jsp
<context root>\private\newrequest\newrequestpage.jsp
<context root>\private\options\optionspage.jsp

Example If subpage is defined in a tabpanel selectedTabParameter attribute, the following tag:

```
<actabpanel:tab key="_scheduled">
```

provides the ability to select this tab by using the following URI:

```
http://<application server>:<port>/iportal/selectjobs.do
?subpage=_scheduled
```

tabBegin

Specifies HTML or JSP code to execute before defining the first tab in a tab panel.

Library	actabpanel
Tag class	com.actuate.activeportal.tags.tabpanel.TabBegin
Attributes	There are no attributes for this tag. Place the desired code as the body of the tag.
Used in	<pre><context root>\private\common\errors\error.jsp <context root>\private\common\sidebar.jsp <context root>\private\newrequest\newrequestpage.jsp <context root>\private\options\optionspage.jsp</pre>
Example	The following example specifies the inclusion of several images to create a border with rounded edges before defining the tabs:

```
<ui:tabBegin>
  <TR>
    <TD>"
      width=8>
    </TD>
    <TD>"
      width=100%>
    </TD>
    <TD>"
      width=8>
    </TD>
  </TR>
</ui:tabBegin>
```

tabEnd

Specifies HTML or JSP code to execute after defining the last tab in a tab panel.

Library	actabpanel
Tag class	com.actuate.activeportal.tags.tabpanel.TabEnd
Attributes	There are no attributes for this tag. Place the desired code as the body of the tag.
Used in	<pre><context root>\private\common\errors\error.jsp <context root>\private\common\sidebar.jsp <context root>\private\newrequest\newrequestpage.jsp</pre>

Example The following example specifies the inclusion of several images to create a border with rounded edges after defining the tabs:

```
<ui:tabEnd>
  <TR>
    <TD><img border=0 height=8
      src=<html:rewrite page="/images/bottom_l_corner.gif"/>
      width=8>
    </TD>
    <TD><img border=0 height=8
      src=<html:rewrite page="/images/horz_stretch.gif"/>
      width=100%>
    </TD>
    <TD><img border=0 height=8
      src=<html:rewrite page="/images/bottom_r_corner.gif"/>
      width=8>
    </TD>
  </TR>
</ui:tabEnd>
```

tabMiddle

Specifies HTML or JSP code to execute for each currently unselected tab.

Library actabpanel

Tag class com.actuate.activeportal.tags.tabpanel.TabMiddle

Attributes There are no attributes for this tag. Place the desired code as the body of the tag.

Used in

```
<context root>\private\common\errors\error.jsp
<context root>\private\common\sidebar.jsp
<context root>\private\jobs\selectjobscontent.jsp
<context root>\private\newrequest\newrequestpage.jsp
<context root>\private\options\optionspage.jsp
```

Example The following example specifies the color, width, alignment, and other attributes of unselected tabs:

```
<ui:tabMiddle>
  <TD bgcolor="#31659C" width=7>&nbsp;</TD>
  <TD bgcolor="#31659C" class="cellSidebar" valign="center"
    nowrap="nowrap">
    <A href="<%= request.getContextPath() %>/{2}"
      class="lnkSidebar">{0}</A>
  </TD>
  <TD bgcolor="#31659C" width=7>&nbsp;</TD>
</ui:tabMiddle>
```

tabMiddleSelected

Specifies HTML or JSP code to execute for the currently selected tab.

Library	actabpanel
Tag class	com.actuate.activeportal.tags.tabpanel.TabMiddleSelected
Attributes	There are no attributes for this tag. Place the desired code as the body of the tag.
Used in	<pre><context root>\private\common\errors\error.jsp <context root>\private\common\sidebar.jsp <context root>\private\jobs\selectjobscontent.jsp <context root>\private\newrequest\newrequestpage.jsp <context root>\private\options\optionspage.jsp</pre>
Example	The following example specifies the color, width, alignment, and other attributes of the selected tab:

```
<ui:tabMiddleSelected>
  <TD bgcolor="#31659C" width=7>&nbsp;  </TD>
  <TD bgcolor="#31659C" class="cellSidebarSelected"
    nowrap="nowrap">
    <A href="<%= request.getContextPath() %>/{2}"
      class="lnkSidebarSelected">{0}</A>
  </TD>
  <TD bgcolor="#31659C" width=7>&nbsp;  </TD>
</ui:tabMiddleSelected>
```

tabPanel

Defines a tab panel and the pages associated with each tab. The tabPanel tag contains other tags from the actabpanel library that specify different parts of the tab panel.

Library	actabpanel
Tag class	com.actuate.activeportal.tags.tabpanel.TabPanelTag
Attributes	Table 7-13 lists and describes the attributes for tabPanel.

Table 7-13 Attributes for tabPanel

Attribute	Required	Description
content Attribute	No	Specifies any HTML attributes to apply to the page part of the HTML table if style=vertical.

(continues)

Table 7-13 Attributes for tabPanel (continued)

Attribute	Required	Description
defaultTab	No	The key of the tab to select if selectedTab is null. If selectedTab and defaultTab are unspecified, the first tab becomes the selected tab.
flush	No	Specifies whether the server should start writing the server response before processing the entire page.
selectedTab	No	Specifies the key of the desired tab. This causes highlighting of the selected tab and display of the page associated with the tab.
selectedTab Parameter	No	Specifies the parameter name that URIs use to specify the key of the desired tab.
style	No	Specifies whether the tab panel is horizontal or vertical.
tabAttribute	No	Specifies any HTML attributes to apply to the tab part of the HTML table if style=vertical.
tableAttribute	No	Specifies any HTML attributes to apply to the nested HTML table containing the tabs.

Used in <context root>\private\common\errors\error.jsp
 <context root>\private\common\sidebar.jsp
 <context root>\private\jobs\selectjobscontent.jsp
 <context root>\private\newrequest\newrequestpage.jsp
 <context root>\private\options\optionspage.jsp

Example The following example creates a tab panel with four tabs. The _completed tab is chosen by default and URLs can specify the tab desired by using subpage=<tab key>:

```
<ui:tabPanel
  selectedTabParameter="subpage" defaultTab="_completed" >
  <ui:tab key="_scheduled">
    <bean:message key="TAB_SCHEDULES"/>
    <ui:content page="scheduledjob.jsp"/>
  </ui:tab>
  <ui:tab key="_pending" >
    <bean:message key="TAB_PENDING"/>
    <ui:content page="pendingjob.jsp"/>
  </ui:tab>
  <ui:tab key="_running" >
    <bean:message key="TAB_RUNNING"/>
    <ui:content page="runningjob.jsp"/>
  </ui:tab>
</ui:tabPanel>
```

```

</ui:tab>
<ui:tab key="_completed" >
    <bean:message key="TAB_COMPLETED"/>
    <ui:content page="completedjob.jsp"/>
</ui:tab>
</ui:tabPanel>

```

tabSeparator

Specifies HTML or JSP code to execute between defining each adjacent pair of tabs.

Library	actabpanel
Tag class	com.actuate.activeportal.tags.tabpanel.TabSeparator
Attributes	There are no attributes for this tag. Place the desired code as the body of the tag.
Used in	<pre> <context root>\private\common\errors\error.jsp <context root>\private\common\sidebar.jsp </pre>
Example	The following example specifies the inclusion of several images to create a dividing line between the tabs:

```

<ui:tabSeparator>
    <TR style="width: 100%">
        <TD colspan=3>
            
                "width=100% height=8 border=0>
        </TD>
    </TR>
    <TR style="width: 100%">
        <TD colspan=3>
            "
                width=100% height=1 border=0>
        </TD>
    </TR>
</ui:tabSeparator>

```

tabSeparator

Actuate Information Console JavaBeans

This chapter contains the following topics:

- Information Console JavaBeans overview
- Information Console JavaBeans package reference
- Information Console JavaBeans class reference
- Information Console UserInfoBean class reference

Information Console JavaBeans overview

This section describes the Information Console JavaBeans. Information Console JavaBeans provide functionality, business logic, and dynamic content to Information Console web applications. Information Console JavaBeans are in `aciportal.jar`, which resides in `<context root>\WEB-INF\lib`.

The Javadoc is provided for the JavaBeans in `<Actuate product root>\iHub2\servletcontainer\mgmtconsole\help\api`. Refer to the Javadoc for a list of JavaBean methods and their arguments.

Information Console JavaBeans package reference

Table 8-1 lists and describes the Actuate packages used in Information Console.

Table 8-1 Information Console packages

Package	Contents
<code>com.actuate.activeportal.beans</code>	JavaBeans that maintain information used by the Action classes.
<code>com.actuate.activeportal.forms</code>	JavaBeans derived from the Jakarta Struts <code>org.apache.struts.action.ActionForm</code> object. These JavaBeans store and validate the request parameters in HTTP requests.
<code>com.actuate.activeportal.list</code>	An interface, <code>IContentList</code> , that defines the behavior of lists of items such as files and channels. Several classes in <code>com.actuate.activeportal.forms</code> use this interface.

Information Console JavaBeans class reference

This section lists and describes the Information Console JavaBean classes by topic.

Channels

Table 8-2 lists and describes Information Console `com.actuate.activeportal.forms` classes that support channels.

Table 8-2 Channel classes

Class	Description
<code>ChannelListActionForm</code>	Provides the list of channels to which the user subscribes or has available.

Table 8-2 Channel classes

Class	Description
GeneralFilter ActionForm	Serves as a base ActionForm for several other ActionForms. Provides methods that handle filters to select which items the Actuate BIRT iHub returns. For example, you can request all folders and only the most recent version of all executable files.
SubscribeChannel ActionForm	Stores a list of channels available to the user, including unsubscribed channels.

Documents

Table 8-3 lists and describes Information Console com.actuate.activeportal.forms classes that support the Document pages.

Table 8-3 Document classes

Class	Description
BrowseFileActionForm	Supports browsing through the available files, including using filters to search.
CreateFolderActionForm	Supports creating a folder in the Encyclopedia volume.
FileFoldersPrivilegeAction Form	Stores information about file and folder access rights, the available users and roles, and so forth. Information Console uses this information to set up file and folder privileges.
FileListActionForm	Retrieves a list of folders or files. This ActionForm supports setting filters specifying characteristics of objects. Stores the most recent list of items returned from iHub.
GeneralFilterActionForm	The base ActionForm for several other ActionForms. Provides methods that handle filters to select which items the iHub returns. For example, you can request all folders and only the most recent version of all executable files.
GetFileDetailsActionForm	Stores the details of a file or folder. AcGetFileDetailsAction gets the details and stores them in this JavaBean.
SearchFilesActionForm	Stores information about the filter set by the user in the Search page. Jakarta Struts uses the filter to retrieve the list of files from the iHub and store them in this form.

General

Table 8-4 describes the Information Console `com.actuate.activeportal.beans` class that supports general functionality.

Table 8-4 General bean class

Class	Description
LinkBean	Generates an HTML link tag using the <code>link</code> , <code>linkAttributes</code> , and <code>text</code> properties. By default, the link class is <code>hyperlink</code> . After setting these properties, use the <code>toString()</code> method to generate an HTML link tag in the following format: <pre>text</pre>

Table 8-5 lists and describes Information Console `com.actuate.activeportal.forms` classes that support general functionality.

Table 8-5 General forms classes

Class	Description
BaseActionForm	The base <code>ActionForm</code> for all other Information Console <code>ActionForms</code> . Provides methods related to postback.
PingActionForm	Stores information used by the ping action. Ping detects the status of Information Console and iHub communication.

Jobs

Table 8-6 lists and describes Information Console `com.actuate.activeportal.forms` classes that support jobs.

Table 8-6 Job classes

Class	Description
GeneralFilterActionForm	Serves as a base <code>ActionForm</code> for several other <code>ActionForms</code> . Provides methods that handle filters to select which items the iHub returns. For example, you can request all folders and only the most recent version of all executable files.
GetJobDetailsActionForm	Stores detail information on jobs. <code>AcGetJobDetailsAction</code> uses this class to store and retrieve the job detail information for display.
JobActionForm	The base <code>ActionForm</code> for <code>SubmitJobActionForm</code> . Stores values used in submitting a job, such as the document, parameters, and schedule.
SelectJobNoticesActionForm	Stores the list of job notices for a channel.

Table 8-6 Job classes

Class	Description
SelectJobsActionForm	Contains the list of job properties for a scheduled, running, pending, or completed job.
SubmitJobActionForm	Contains the information for submitting a job from the requester page. This class extends JobActionForm.

Skins

Table 8-7 lists and describes Information Console `com.actuate.activeportal.beans` classes that support skins.

Table 8-7 Skin bean classes

Class	Description
GroupBean	Stores lists of all images, colors, fonts, and styles for a skin. Each list is a list of SkinBean objects.
SkinBean	Stores information about an image, style, color, or font. The information for this JavaBean comes from a <code><Style></code> or <code><Image></code> tag in the <code>skin.config</code> file for a skin. Access this information using the <code>getStyle()</code> or <code>getImage()</code> methods. SkinBeans are grouped into GroupBeans for each skin.
SkinManagerInfoBean	Stores access information about a skin. Used by the <code>SkinManagerActionForm</code> .

Table 8-8 lists and describes Information Console `com.actuate.activeportal.forms` classes that support skins.

Table 8-8 Skin form classes

Class	Description
FileUploadActionForm	Uploads images during skin customization and stores an object representation of the uploaded file. It uses Jakarta Struts <code>org.apache.struts.upload.FormFile</code> to handle file upload. The file is saved in a temporary folder on the server.
SkinEditorActionForm	Stores all the information about the various groups defined in the <code>skin.config</code> file. When an administrator edits a skin, Information Console loads the <code>skin.config</code> file and represents its contents as a <code>SkinConfig</code> object. Changes to the skin's images, color, and fonts are stored in GroupBeans for a skin.
SkinManagerActionForm	Stores the list of available skins. Use the <code>getSkin()</code> method to get the list of skins as a <code>Vector</code> of <code>SkinManagerInfoBean</code> . This form supports adding, cloning, and deleting skins.

Users

Table 8-9 lists and describes Information Console `com.actuate.activeportal.beans` classes that support handling users.

Table 8-9 User bean classes

Class	Description
FeatureOptionsBean	Stores the features available to the current user. It contains Information Console functionality levels and reporting features on the iHub the user is using. Access this class using <code>UserInfoBean.getFeatureBean()</code> .
ProfileBean	Stores the user profile settings obtained from the iHub. Access this class by using <code>UserInfoBean.getProfile()</code> .
UserAgentBean	Detects what kind of browser the user is using from the HTTP header user-agent. After instantiating this <code>JavaBean</code> , you must call <code>setRequest(HttpServletRequest request)</code> . Get the browser type by calling <code>isIE()</code> , <code>isNS4()</code> , and <code>isNS6()</code> methods.
UserInfoBean	Contains information about the user, such as the user's Encyclopedia volume name, iHub URL, preferred skin, and authentication ID assigned by the iHub. Several methods also affect the display and highlighting of features.

Table 8-10 lists and describes Information Console `com.actuate.activeportal.forms` classes that support handling users.

Table 8-10 User form classes

Class	Description
LoginForm	Stores information about the user ID, server URL, volume, and other information specified during login.
UserOptionsActionForm	Stores the selected choices on the options page, including the skin, view, experience level, and e-mail ID. This form supports changing these options.

Information Console `UserInfoBean` class reference

Table 8-11 lists and describes the methods other than `set` methods available in the Information Console `com.actuate.activeportal.beans.UserInfoBean` class.

Table 8-11 UserInfoBean methods

Method	Description
getAcLocale()	Gets the AcLocale object specifying the Actuate locale for the current user.
getAdminRights()	Gets the administrator rights of the current user. If the user is not an administrator or operator, this method returns null. An administrator or application sets these rights when creating a user.
getAuthid()	Gets a String containing the authentication ID returned by the iHub for this user during login. Use this authentication ID in IDAPI calls.
getCurrentfolder()	Gets a string containing the name of the most recent folder accessed by the user.
getDefaultServerURL()	Gets a string for the URL to use for the default server for the user.
getDefaultVolume()	Gets the volume name from the VOLUME_DEFAULT tag in <context root>\WEB-INF\web.xml.
getFeatureOptionsBean()	Gets a JavaBean that stores the features and iHub options that are available to the current user.
getFeatures()	Gets a list of all features defined in the functionality-level.config file.
getFilter()	Gets a string containing the filter the user most recently typed into the search field of the Documents page. If the user has not typed a filter, this method returns null.
getHomefolder()	Gets a string specifying the user's home folder. An administrator or application sets this value when creating a user.
getIportalid()	Gets a string specifying the Information Console session id.
getLocale()	Gets the current login user's java.util.Locale object.
getMaxJobPriority()	Gets the maximum job priority permitted for this user. An administrator or application sets this value when creating a user.
getOnlylatest()	Gets the string "true" if the filter on the Documents page specifies showing only the most recent version of each file.
getPassword()	Gets a string containing the user's password.

(continues)

Table 8-11 UserInfoBean methods (continued)

Method	Description
<code>getProfile()</code>	Gets the ProfileBean. This JavaBean stores information about the user's settings on the Information Console options page. This information includes current skin, view, experience level, and so on.
<code>getProperty(java.lang.String name)</code>	Gets a string containing the value of a custom property having the name passed as a parameter. Create custom properties and set their values using <code>setProperty()</code> .
<code>getRepositoryType()</code>	Gets a string specifying the type of repository that the user is accessing as: workgroup: local file system enterprise: an Encyclopedia volume
<code>getRoleNames()</code>	Gets an array of strings containing a list of the user's feature roles, such as Actuate Information Console Intermediate, Actuate Information Console Advanced, or Actuate Information Console Administrator.
<code>getServerurl()</code>	Gets the URL of the server to which the current user is logged in. This URL includes the protocol and the port. For example: <code>http://localhost:9000</code> .
<code>getShowdocuments()</code>	Gets the string "true" if the filter on the Documents page specifies including documents.
<code>getShowexecutables()</code>	Gets the string "true" if the filter on the Documents page specifies including executable files.
<code>getShowfolders()</code>	Gets the string "true" if the filter on the Documents page specifies including folders.
<code>getSideBarFeatures()</code>	Gets the list of features available to this user on the side menu, tabs, the tree, or equivalent structure. Some features, such as customization, are not part of this set.
<code>getSidebarSelected()</code>	Gets the URL for the feature selected on the side menu, tab, tree, or equivalent structure. This method is used to highlight a feature in the sidebar.
<code>getSkinConfig()</code>	Gets the SkinConfig object for the user's current skin. The SkinConfig object contains all information defined for the skin.
<code>getSkinName()</code>	Gets a string containing the name of the skin used by the user.
<code>getSubfeatures()</code>	Gets a collection containing a list of all subFeatures defined in <code><context root>\WEB-INF\functionality-level.config</code> .
<code>getSystemname()</code>	Gets a string containing the name of the iHub machine.

Table 8-11 UserInfoBean methods (continued)

Method	Description
getTimezone()	Gets the AcTimeZone object specifying the time zone for the current user.
getUserAgent()	Gets the UserAgentBean object for the user. UserAgentBean detects the user's browser type.
getUserid()	Gets the userID of the current user.
getView()	Gets the string specifying the current view for this user.
getVolume()	Gets the string specifying the Encyclopedia volume that the user is accessing.
init()	Initializes the UserInfoBean members.
isAlwaysGetFolderList()	Returns True if the Documents page should always show the folder list, even if it is not selected on the filter.
isHomeFolderSet()	Returns True if the user has a home folder specified in the Encyclopedia volume.
isShowFilters()	Returns True if the filter panel is shown for all lists of documents, jobs, and channels.
isViewInNewBrowserWindow()	Returns True if the report viewer is specified to launch in a new browser window.
toString()	Returns a string representation of the object.

UserInfoBean calls set methods when the user logs in to set the values that the get methods return. Typically, your application should not call the set methods as the bean would then be inconsistent with the information stored in the repository or external security application. These set methods only change the values in the bean, so the results of the calls are not deterministic.

Table 8-12 lists and describes set methods that are available in the Information Console com.actuate.activeportal.beans.UserInfoBean class.

Table 8-12 UserInfoBean set methods

Method	Description
setAcLocale(com.actuate.reportcast.utils.AcLocale acLocale)	Sets the Actuate locale for the current user with the specified AcLocale object. Also changes the Java locale.
setAlwaysGetFolderList(boolean b)	Set to True if the Documents page should always show the folder list, even if it is not selected on the filter.

(continues)

Table 8-12 UserInfoBean set methods (continued)

Method	Description
setAuthid(java.lang.String authid)	Sets the authentication ID to the string passed in as a parameter. The authentication ID is returned by the Actuate BIRT iHub and set for the user during login. Use getAuthid() to use this authentication ID in IDAPI calls.
setCurrentfolder(java.lang.String currentfolder)	Sets the string specifying the most recent folder name accessed by the user.
setDefaultServerURL(java.lang.String defaultServerURL)	Sets the URL to use as a default value for users.
setDefaultVolume(java.lang.String defaultVolume)	Sets the volume to use if no volume name is specified by the URL in the request. By default, Information Console sets the default volume to the value in the VOLUME_DEFAULT tag in <context root>\WEB-INF\web.xml.
setFeatureOptions (FeatureOptionsBean featureOptionsBean)	Sets a list of all Information Console features and iHub options that are available to the current user.
setFilter(java.lang.String filter)	Sets the string specifying the filter to use as a default value in the Documents page. Information Console sets this String to the filter that the user most recently typed into the search field of the Documents page.
setHomefolder(java.lang.String string)	Sets the string specifying the user's home folder. An administrator or application sets this value when creating a user.
setMaxJobPriority(int priority)	Sets the maximum job priority permitted for this user. An administrator sets this value for a user.
setOnlylatest(java.lang.String onlylatest)	Sets value to indicate if only latest version of the documents are to be displayed in the file folder list. "true" sets Information Console to show only the most recent version of each file.
setPassword(java.lang.String password)	Sets the password to the value of the string passed as a parameter.
setProfile()	Sets the ProfileBean. This JavaBean stores information about the user's settings on the Information Console options page. This information includes current skin, view, experience level, and so on.
setProperty(java.lang.String name, java.lang.String value)	Sets the value of a custom property. Create custom properties and set their values using this method. The parameters are the name of the custom property and the value to set.

Table 8-12 UserInfoBean set methods (continued)

Method	Description
setRoleNames(java.lang. .String[] strings[])	Sets a list of the user's feature roles, such as Active Portal Intermediate, Active Portal Advanced, or Active Portal Administrator.
setServerurl(java.lang. .String surl)	Sets the server URL currently used by the user. This URL includes the protocol and the port, for example: <code>http://localhost:9000</code> .
setShowdocuments(java. .lang.String showdocuments)	Sets the value to indicate if documents are to be displayed in the file folder list. "true" sets Information Console to display documents.
setShowexecutables(java. .lang.String showexecutables)	Sets the value to indicate if executables are to be displayed in the file folder list. "true" sets Information Console to display executables.
setShowFilters(boolean showFilters)	Set to True to specify that Information Console display the filter panel for all pages showing lists of files, jobs, or channels.
setShowfolders(java.lang. .String showfolders)	Sets the value to indicate if folders are to be displayed in the file folder list. "true" sets Information Console to display folders.
setSideBarFeatures(com. .actuate.activeportal. .functionality.config. .Feature[] feature)	Sets the list of features available to this user on the side menu, tabs, the tree, or equivalent structure. This list is a subset of the features available to the user.
setSidebarSelected(java.lang. .String sideBarSelected)	Sets the feature highlighted on the side menu, tab, tree, or equivalent structure. To highlight a feature, pass a string containing the URI invoked by the feature. To not highlight any features, pass a string, such as "No highlighting", that does not match the URI for any feature in the side menu. By default, Information Console highlights the Documents feature.
setSkinConfig(com.actuate. .activeportal.skin. .SkinConfig config)	Sets the SkinConfig object for the user's current skin. The SkinConfig object contains all information defined for the skin.
setSkinName(java.lang. .String string)	Sets the name of the skin used by the user.
setSystemname(java.lang. .String systemName)	Sets the iHub system name to the value of the string parameter.

(continues)

Table 8-12 UserInfoBean set methods (continued)

Method	Description
setTimezone(com.actuate .reportcast.utils .AcTimeZone timezone)	Sets the AcTimeZone object specifying the user's time zone.
setUserAgent (UserAgentBean userAgent)	Sets the UserAgentBean for this user. The UserAgentBean specifies the user's browser type.
setUserid(java.lang .String userid)	Sets the user ID for the user.
setView(java.lang.String string)	Sets the current view for the user. The string contains the name of the constant for the desired view. The available constants are: <ul style="list-style-type: none">■ AcConstants.VIEW_CATEGORY■ AcConstants.VIEW_LIST■ AcConstants.VIEW_DETAIL■ AcConstants.VIEW_ICON
setViewInNewBrowser Window(boolean _newWindow)	Set to True to specify that the report viewer launch in a new browser window.
setVolume(java.lang .String volume)	Sets the value of the string specifying the name of the Encyclopedia volume the user is accessing.

Using Actuate Information Console security

This chapter contains the following topics:

- About Actuate Information Console security
- Protecting corporate data
- Understanding the authentication process
- Creating a custom security adapter
- Creating an upload security adapter

About Actuate Information Console security

A reporting web application is accessible to any user who has a web browser and the URI for the application. This chapter discusses the Actuate Information Console security features and how to use them to:

- Ensure that users access only those objects in the Encyclopedia volume for which they have permission.
- Protect sensitive reports.

The types of security you can provide for Information Console are:

- Default user authentication. Use the default Information Console and Actuate BIRT iHub facilities to ensure that users access only those reports and other Encyclopedia volume items for which they have permission.
- User authentication using the Information Console Security Extension (IPSE). Use IPSE to customize and control the user login and authentication process. For details about implementing custom user authentication, see “Creating a custom security adapter,” later in this chapter.

Protecting corporate data

iHub provides a structured content generation solution for web applications. Deploying Actuate applications developed for the internet, such as Information Console, requires planning for network security.

Internet applications support access to information within an organization from outside that organization. Because the organization’s internal network is connected to the internet, there is the risk of unauthorized access to the corporate network and to the data that resides on that network.

Organizations use one or a combination of the technologies described in the following sections to prevent unauthorized access to the corporate network and protect authentication transactions from intrusion.

Protecting corporate data using firewalls

Typically companies use firewalls to prevent unauthorized access to corporate networks and data. A firewall is a system or group of systems that restrict access between two networks, such as an organization’s internal network and the internet. Firewalls keep unauthorized users out. As a result, firewalls prevent damage caused by malicious programs such as worms and viruses from spreading to other parts of your network. At the same time, firewalls allow legitimate business to tunnel through the firewall and be efficiently conducted on your network.

Firewalls can be used to restrict access between two internal networks, for example, the accounting and engineering networks. Security teams configure firewalls to allow traffic using specific protocols, such as HTTP, over specific network addresses and ports. Be sure that your firewall allows access for the Information Console and iHub ports. For more information about the Actuate ports, see *Configuring BIRT iHub*.

Protecting corporate data using Network Address Translation

Companies also use Network Address Translation (NAT). NAT routers and software support private networks using unregistered, private IP (Internet Protocol) addresses to connect to the internet.

Protecting corporate data using proxy servers

Proxy servers, specialized web servers or hardware that operate on or behind a firewall, improve efficient use of network bandwidth and offer enhanced network security. For more information about proxy servers and Information Console, see Chapter 1, “Introducing Actuate Information Console.”

Understanding the authentication process

The authentication process involves the following steps, in this order:

- A user or client makes a request by choosing a link on an Information Console page or by typing an Actuate Information Console URI in a web browser. The Information Console application processes the request.
- Information Console checks the URI for the `forceLogin` parameter. If the `forceLogin` parameter is set to “true” in the URI, the application activates the Information Console Login page, even if the user has already logged in. If `forceLogin` is set to “false” or does not appear, the request process continues. For details about the `forceLogin` URI parameter, see “Common URI parameters” in Chapter 4, “Actuate Information Console URIs.”
- Information Console authenticates the user for the Encyclopedia volume. If the login information is invalid, the login screen appears in the browser.

If a custom security adapter parameter is set in the `web.xml` file, Information Console attempts to load the custom security adapter class. If the class loads successfully, the following steps occur:

- Information Console calls the custom security adapter’s `authenticate()` method with the parameters that the browser sent.
- The `authenticate()` method performs the custom validation.

- Information Console calls the required `getUserName()`, `getPassword()`, and `getVolumeProfile()` methods to retrieve the user information needed by the iHub.
- Optionally, Information Console calls the `getExtendedCredentials()` method. If this method returns null, there are no extended credentials to send to the iHub.
- Information Console now has all the information that it requires for connecting to the iHub. Information Console creates the necessary SOAP message for connecting to the iHub and sends a login request.

Information Console uses the default Volume Profile setting if the server, volume, or volume profile

Creating a custom security adapter

The Information Console security adapter enables other applications to authenticate users and log in to the Information Console application, for example, by using a URL. A custom security adapter can define alternate authentication requirements. In this way, an Information Console security adapter establishes an additional layer of logic to the existing Information Console authentication, as shown in Figure 9-1.

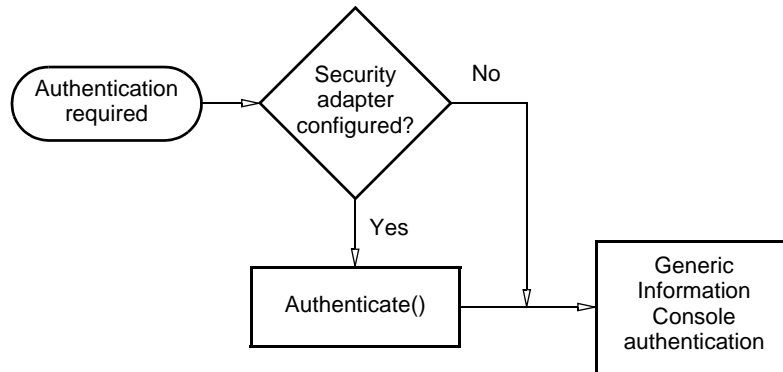


Figure 9-1 Information Console authentication system

A user cannot update their password from Information Console if a custom security adapter class is set. In this way, Information Console prevents conflicts between the user's current password and the security system that is used to verify passwords.

To create a custom security adapter, perform the following steps:

- Ensure that your application can access the IPSE Java classes.

- Create a java class that implements the custom security adapter class for IPSE.
- Deploy the Custom Security Adapter to Information Console.

Accessing the IPSE Java classes

The Information Console library, `com.actuate.iportal.jar`, contains the IPSE Java classes. This library is located in the `lib` subdirectory in the Information Console installation. The class, `com.actuate.iportal.security.iPortalSecurityAdapter`, in this library provides the framework for custom authentication. A custom security adapter providing an IPSE implementation extends this class.

Specifically, the JRE needs to access the following jars:

- `<context root>\WEB-INF\lib\com.actuate.iportal.jar`
- `<context root>\WEB-INF\lib\org.apache.xerces_<version>.jar`
- `<context root>\WEB-INF\lib\com.actuate.webcommon.jar`
- `<iPortal installation directory>\lib\servlet-api.jar`
- `<iPortal installation directory>\lib\jsp-api.jar`

Creating a custom security adapter class

Extend the `iPortalSecurityAdapter` class to customize authentication. The `iPortalSecurityExtension` requires access to the following libraries:

- `javax.servlet.http.*`
- `com.actuate.iportal.security.iPortalSecurityAdapter`

`iPortalSecurityAdapter` provides a set of empty methods. Extend this class and override any of the methods to provide custom IPSE authentication. To establish a secure session with Information Console using a custom security adapter, the following methods are required:

- A constructor
- `authenticate()`
- `getPassword()`
- `getUserName()`

The login module of Information Console calls methods in the custom security class to perform authentication and to retrieve login credentials to pass to `iHub`. The `authenticate()` method returns a boolean value to indicate whether the login credentials provided are acceptable. The getter methods return the credentials that `iHub` requires. Each user name and password must correspond to an authentic user account on the volume configured by the volume profile. If a volume profile isn't set by the security adapter, authentication uses the default

volume profile configuration. For example, to support a URL that authenticates using a single parameter, code, override `authenticate()` to retrieve the parameter from the `HttpServletRequest` and set the user name, password, and volumeProfile as in the following class:

```
import javax.servlet.http.*;
import com.actuate.iportal.security.iPortalSecurityAdapter;

public class SecurityCode extends
    com.actuate.iportal.security.iPortalSecurityAdapter {
    private String volumeProfile = "CustomAccess";
    private String userName = null;
    private String password = null;
    public SecurityCode() {}

    public boolean authenticate(
        HttpServletRequest httpServletRequest) {
        String param = httpServletRequest.getParameter("code");
        boolean secured = true;
        if ("12345".equalsIgnoreCase( param )) {
            userName = "user1";
            password = "user1";
        }
        else if ("abc".equalsIgnoreCase( param )) {
            userName = "BasicUser";
            password = "";
        }
        else {
            secured = false;
        }
        return secured;
    }
    public String getUserName() { return userName; }
    public String getPassword() { return password; }
    public String getVolumeProfile() { return volumeProfile; }
}
```

If there is a user "user1" with the password "user1" on the volume configured by the volume profile "CustomAccess," a valid URL that authenticates user1 using this security adapter is as follows:

`http://localhost:8700/iportal/getfolderitems.do?code=12345`

Deploying a custom security adapter

To deploy a custom security adapter, the Information Console application must have access to the class compressed into a JAR file. To meet this requirement, compile the class, compress it into a JAR, and move it into the `<context root>\WEB-INF\lib` directory for your Information Console application. Then, add the

class's name as the value for the SECURITY_ADAPTER_CLASS parameter in <context root>\WEB-INF\web.xml. Finally, restart the application service running Information Console to activate this change.

How to deploy a custom security adapter to Information Console

- 1 Compile the IPSE application. Use a command similar to this one in a console window:

```
javac SecurityCode.java
```

- 2 Create a JAR file to contain the IPSE application. Use a command similar to this one in a console window:

```
jar cvf SecurityCode.jar SecurityCode.class
```

- 3 Using Windows Explorer, copy SecurityCode.jar to this directory:

```
<your application context root>\WEB-INF\lib
```

- 4 Using a UTF-8 compliant code editor, open the following file:

```
<your application context root>\WEB-INF\web.xml
```

- 5 Navigate to the parameter name SECURITY_ADAPTER_CLASS.

- 6 Change the param-value parameter of the SECURITY_ADAPTER_CLASS to the fully qualified class name for the security adapter class. Use an entry similar to this one:

```
<param-name>SECURITY_ADAPTER_CLASS</param-name>  
<param-value>SecurityCode</param-value>
```

- 7 Save and close web.xml.

- 8 Restart the application server running Information Console. For the default installation, restart the Actuate 11 Apache Tomcat for Information Console Service.

Understanding the security adapter class

To implement a custom security adapter, create a class that extends `com.actuate.iportal.security.iPortalSecurityAdapter`. This class contains the following methods.

authenticate()

Syntax	<code>boolean authenticate(javax.servlet.http.HttpServletRequest request)</code>
Parameters	request The request parameter sent from the Information Console web application.
Description	Required method that evaluates the current user's security credentials. The Login module calls <code>authenticate()</code> to validate the current user's security credentials. If <code>authenticate()</code> returns <code>False</code> , the user is redirected to the login page.

Returns True for successful credential evaluation and False otherwise.

Throws An AuthenticationException indicating the reason for the failure, if credential evaluation is not successful.

getExtendedCredentials()

Syntax byte[] getExtendedCredentials()

Description Retrieves the current user's extended security credentials.

Returns A byte array representing any extended credentials for the iHub to use to authenticate the user, or null if there are no extended credentials to evaluate.

getPassword()

Syntax String getPassword()

Description Required method that retrieves the current user's password. The Login module calls getPassword() and uses the password to establish a connection to the iHub and to access the Encyclopedia volume.

Returns A string that is the password to use to establish the connection.

getRepositoryType()

Syntax String getServerUrl()

Description Retrieves the repository type. The Login module calls this method to check the repository type. Alternatively, provide isEnterprise().

Returns A string that indicates the repository type. The repository type for iHub is enterprise.

getRunAsUser()

Syntax String getRunAsUser()

Description Retrieves the runAs setting if the runAs is enabled. The Login module calls this method to retrieve the user name used for a run as operation.

Returns A string containing the user name that corresponds to the runAs user setting.

getServerUrl()

Syntax String getServerUrl()

Description Retrieves the URL of the server to which the current user connects. The Login module calls getServerURL().

Returns A string containing the URL for the iHub currently connected.

getUserHomeFolder()

- Syntax** String getUserHomeFolder()
- Description** Retrieves the current user's home folder. The Login module calls getUserHomeFolder() to access the user's files.
- Returns** A string that is the user's home folder. It is null if there is no home folder for the user.

getUserName()

- Syntax** String getUserName()
- Description** Required method that retrieves the current user's login name. The Login module calls getUserName() to establish a connection to the iHub and to access the Encyclopedia volume.
- Returns** A string containing the user name that the iHub recognizes.

getVolume()

- Syntax** String getVolume()
- Description** Retrieves the volume to which the current user connects. The Login module calls getVolume() to retrieve the name of the Encyclopedia volume to which the user wishes to connect.
- Returns** A string containing the domain and volume name for the Encyclopedia volume to which the user connects to through the iHub. If null, the iHub connects to the default volume, read from the DEFAULT_VOLUME parameter in the Information Console web.xml file.

getVolumeProfile()

- Syntax** String getVolumeProfile()
- Description** Required method that retrieves the volume profile to which the current user connects. The Login module calls getVolumeProfile() to retrieve the name of the volume profile to which the user wishes to connect.
- Returns** A string containing the server profile name for the Encyclopedia volume to which the user connects through the iHub.

isEnterprise()

- Syntax** boolean isEnterprise()
- Description** Evaluates whether the user connects to an Encyclopedia volume. The Login module calls isEnterprise() to determine whether to use an Encyclopedia volume repository.
- Returns** True.

Creating an upload security adapter

The default security for Information Console upload functionality checks a file's type against the value of the `UPLOAD_FILE_TYPE_LIST` parameter in `web.xml`. The Information Console upload security adapter provides additional external verification features for the file upload feature using a Java interface, `com.actuate.iportal.security.IUploadSecurityAdapter`.

Information Console upload security adapter establishes an additional layer of logic to the existing Information Console authentication, as shown in Figure 9-2.

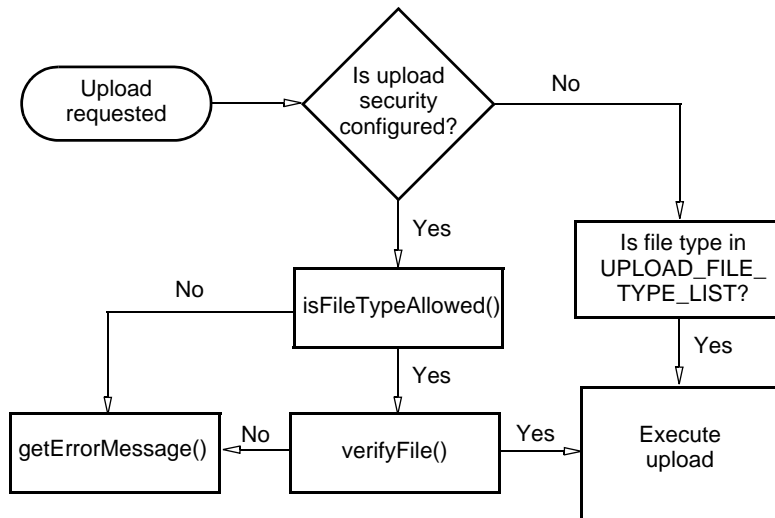


Figure 9-2 Information Console upload security system

If an upload security adapter is configured, Information Console calls `isFileTypeAllowed` to check whether the file's file type is allowed. If so, then it calls `verifyFile` to perform any additional verification steps. If either `isFileTypeAllowed` or `verifyFile` returns false, Information Console displays an error message supplied by `getErrorMessage`, or a generic message if `getErrorMessage` returns null.

To create an upload security adapter, perform the following steps:

- Ensure that your application can access the necessary Java classes.
- Create a Java class that implements the upload security adapter interface.
- Deploy the upload security adapter class to Information Console.

Accessing the necessary Java classes

The Information Console library, `com.actuate.iportal.jar`, contains the security extension Java classes. This library is located in the `lib` subdirectory of the Information Console installation. The upload security adapter interface, `com.actuate.iportal.security.IUploadSecurityAdapter`, in this library provides the framework for additional upload security. A valid upload security adapter implements this interface.

Specifically, the JRE needs to access the following jars:

- `<context root>\WEB-INF\lib\com.actuate.iportal.jar`
- `<context root>\WEB-INF\lib\org.apache.xerces_2.9.0.v201005080400.jar`
- `<iPortal installation directory>\lib\servlet-api.jar`
- `<iPortal installation directory>\lib\jsp-api.jar`

Creating a custom security adapter class

Implement the upload security adapter interface to customize file verification. The upload security adapter requires access to the following libraries:

- `javax.servlet.http.HttpServletRequest`
- `javax.servlet.ServletContext`
- `com.actuate.iportal.security`

To process a secure upload request from Information Console using an upload security adapter, the following methods are required:

- `getErrorMessage()`
- `isFileTypeAllowed()`
- `verifyFile()`

For example, to prevent any file type except plain text (`.txt`) from being uploaded, implement `txt` as the only valid file type for `isFileTypeAllowed`, as in the following class:

```
package com.actuate.iportal.security;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;

public class SecureUpload implements IUploadSecurityAdapter {

    public boolean isFileTypeAllowed( HttpServletRequest request,
        String fileType ) {
        if ( fileType == null ) return false;
        if ( fileType.toLowerCase().trim().equals("txt") ) return true;
    }
}
```

```

        else return false;
    }

    public boolean verifyFile(HttpServletRequest request, String
        fileName, String dstFolder){
        return true;
    }

    public String getErrorMessage(HttpServletRequest request){
        String message = "Only plain text (.txt) files are permitted.";
        return message;
    }
}

```

When the upload security adapter requires file validation, Information Console copies the file temporarily into the directory specified by TEMP_FOLDER_LOCATION parameter in web.xml.

Deploying an upload security adapter

To deploy an upload security adapter, the Information Console application must have access to the class compressed into a JAR file. To meet this requirement, compile the class, compress it into a JAR, and move it into the <context root>\WEB-INF\lib directory for your Information Console application. Then, add the class's name as the value for the UPLOAD_SECURITY_ADAPTER parameter in <context root>\WEB-INF\web.xml. Finally, restart the application service running Information Console to activate this change.

How to deploy an upload security adapter to Information Console

- 1 Compile the Upload security application. Use a command similar to this one in a console window:

```
javac SecureUpload.java
```

- 2 Create a JAR file to contain the upload security application. Use a command similar to this one in a console window:

```
jar cvf SecureUpload.jar SecureUpload.class
```

- 3 Using Windows Explorer, copy SecureUpload.jar to this directory:

```
<your application context root>\WEB-INF\lib
```

- 4 Using a UTF-8 compliant code editor, open the following file:

```
<your application context root>\WEB-INF\web.xml
```

- 5 Navigate to the parameter name UPLOAD_SECURITY_ADAPTER.

- 6 Change the param-value parameter of the UPLOAD_SECURITY_ADAPTER to the fully-qualified class name for the upload security adapter class. Use an entry similar to this one:

```
<param-name>UPLOAD_SECURITY_ADAPTER</param-name>  
<param-value>SecureUpload</param-value>
```

- 7 Save and close web.xml.
- 8 Restart the application server running Information Console. For the default installation, restart the Actuate 11 Apache Tomcat for Information Console Service.

Understanding the upload security adapter interface

To implement a custom upload security adapter, create a class that implements the `com.actuate.iportal.security.IUploadSecurityAdapter` interface. This interface defines the following methods.

getErrorMessage()

- Syntax** String getErrorMessage(javax.servlet.http.HttpServletRequest request)
- Parameters** **request**
The request parameter sent from the Information Console web application.
- Description** A method that returns a custom error string when either `isFileTypeAllowed` or `verifyFile` returns False.
- Returns** String. An error message. If null, Information Console displays a generic default error message.

isFileTypeAllowed()

- Syntax** boolean isFileTypeAllowed(javax.servlet.http.HttpServletRequest request, string fileType)
- Parameters** **request**
The request parameter sent from the Information Console web application.
- fileType**
String. The type of the upload file, as determined by file extension.
- Description** A required method used to do additional validation of the upload file.
- Returns** Boolean. True for an allowed file type and False otherwise.

verifyFile()

- Syntax** boolean verifyFile(javax.servlet.http.HttpServletRequest request, string filePath, string dstPath)

Parameters	request The request parameter sent from the Information Console web application.
	filePath String. The path of the file stored on Information Console. The default location is the directory specified by TEMP_FOLDER_LOCATION parameter in web.xml.
	dstPath String. The repository path to which the upload sends the file.
Description	A required method used to do additional validation of the upload file.
Returns	Boolean. True for successful file validation and False otherwise.

Customizing Information Console online help

This chapter contains the following topics:

- About Actuate Information Console online help files
- Using a custom help location
- Creating a localized help collection
- Customizing icons, links, and the company logo
- Changing help content

About Actuate Information Console online help files

Actuate provides Information Console online help using the internet by default. To customize online help for Information Console, install the documentation on the local server and switch the help location for Information Console to the local server. Then, customize the online help as needed.

How to switch the help location for Information Console for Windows

Switching the help location is required for any customization task.

- 1 Extract the contents of the win-l10nandonline documentation_iHub2.zip file and run ActuateLocalizationandOnlineDocumentation.exe. Use the default settings for documentation installation to install the help and PDF files for all installed Actuate products.
- 2 From the Start menu, choose Programs→Actuate→Switch Help Location.
- 3 In Switch Help Location, select Use local help, as shown in Figure 10-1.

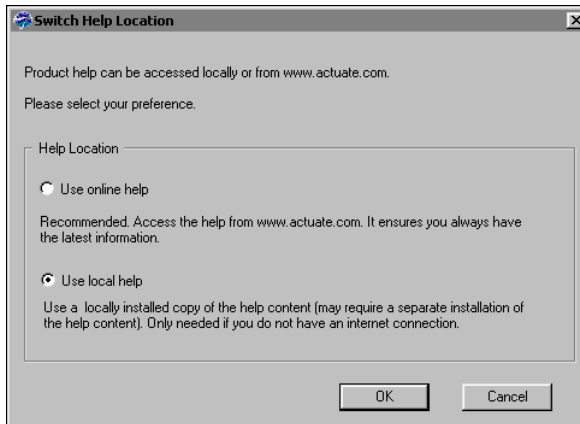


Figure 10-1 Switching the help location for Information Console
Choose OK.

- 4 Restart the service for Information Console. For a stand-alone application, this service is Actuate Apache Tomcat for BIRT iHub 2 Information Console service.

Understanding the help directory structure

The local Information Console help files are grouped into directories under the context root for Information Console, which is the home directory in which the Actuate product resides. For example, the default context root for Information Console installed as a component of iHub on Windows systems is

<Actuate home>\iHub2\servletcontainer\iportal and on Linux systems is <Actuate home>/iHub2/servletcontainer/iportal. The localized help directory under the context root is the container for the help implementation:

<context root>\help

Figure 10-2 illustrates the Information Console help directory structure.

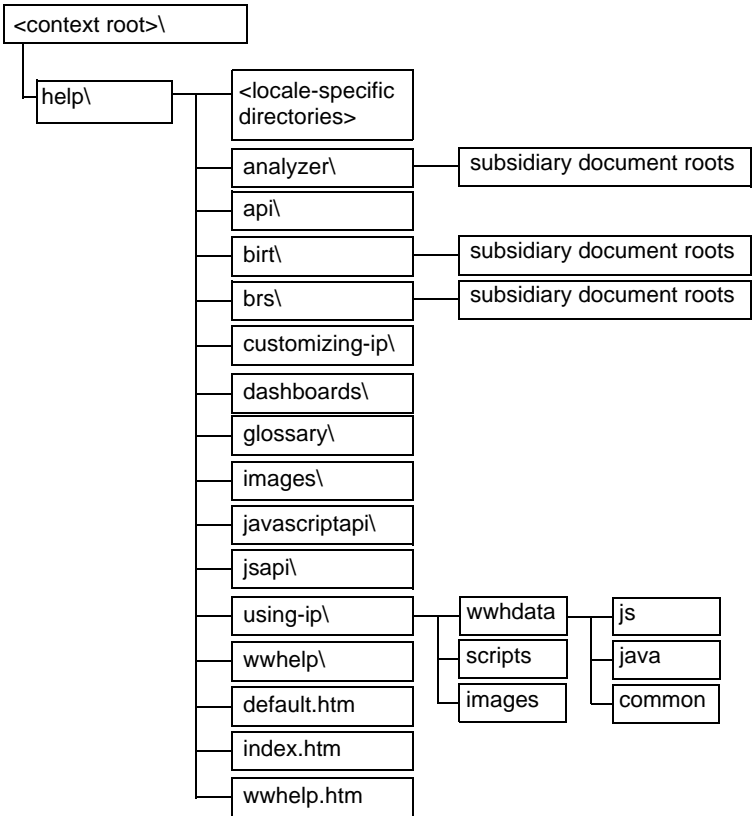


Figure 10-2 Information Console help directory structure

Actuate uses JavaScript (.js) and HTML (.html) files to implement Information Console help. The files that support top-level help styles and images reside in the wwhelp directory. Files that support help content pages and help navigation reside in a document root directory. A document root contains the help files for a specific top-level help topic, such as dashboards or glossary.

Understanding a help collection

The wwhelp directory contains files that support grouping multiple document roots into a collection. If you open the help using index.htm, the table of contents frame displays the top-level help topics, as shown in Figure 10-3.

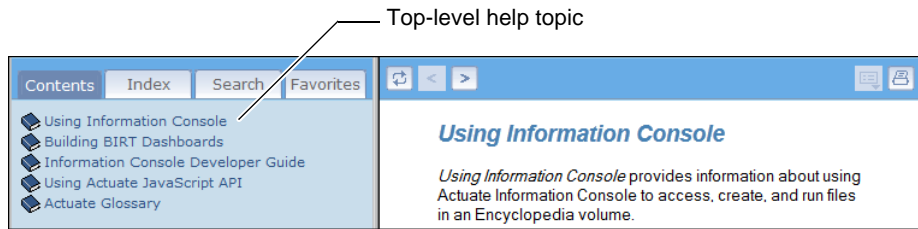


Figure 10-3 Appearance of top-level help topics

A collection has a one-to-one correlation between each top-level help topic and a document root. Each top-level help topic represents a complete book. Table 10-1 lists these applications and the directory containing the corresponding help collection.

Table 10-1 Applications and help collection directories

Application	Directory
Using Information Console	using-ip
Building BIRT Dashboards	dashboards
Information Console Developer Guide	customizing-ip
Using Actuate JavaScript API	javascriptapi
Actuate Glossary	glossary

The help directory contains subdirectories that provide the help collections for applications launched by Information Console. Table 10-2 lists each document root and its corresponding top-level help topic.

Table 10-2 Top-level help topics

Help topic	Document root
Actuate BIRT Viewer and Interactive Viewer	birt
BIRT Data Analyzer	analyzer
BIRT Studio	brs

Understanding a document root

The content files for a top-level help topic reside in a corresponding document root. For example, the using-ip document root contains iPusing-intro.2.1.html, iPusing-intro.2.2.html, and so on. These files are the content files for the help. Each document root also contains an index.html file. Opening this file displays the topic and content files for the book.

Within each document root is a wwldata\common directory that contains the JavaScript files that organize help content and that link the help files to the

application. Table 10-3 lists and describes the customizable <document root>\wwhdata\common contents.

Table 10-3 Help content management files

File	Purpose
files.js	Lists the content files to be used and in what order
title.js	Specifies the title for the browser window and the top-level table of contents text
topics.js	Designates the targets for context-sensitive help keys the Information Console emits

Within each document root, a wwhdata\js directory contains JavaScript files that organize the navigation frame. This frame includes the table of contents (TOC), index, and search frames. Table 10-4 lists and describes the customizable <document root>\wwhdata\js contents.

Table 10-4 Help navigation files

File	Purpose
index.js	Organizes the index links and hierarchy
search.js	Designates specific search values and priority
toc.js	Specifies the table of contents frame hierarchy, linking behavior, and text

Understanding context-sensitive help

The Information Console application links to help files using wwhelp.html located in <context root>\help. Typically, links that activate this context-sensitive help are in the Information Console application, as shown in Figure 10-4.

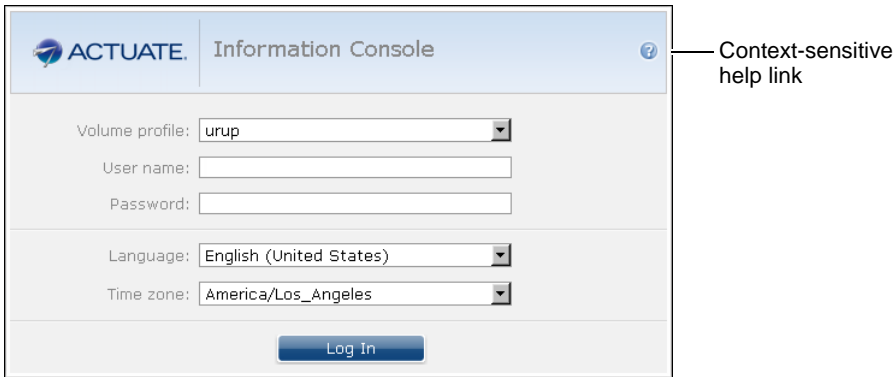


Figure 10-4 Information Console help link for login page

These links in the Information Console emit a URL for the `wwhelp.html` file and append two parameters to that URL, context and topic. The URL looks like following example:

```
http://host:8700/portal/help/wwhelp.htm#context=UserConsole
&topic=Dashboard
```

- `host` is the name of the web server serving online help.
- `8700` is the port number for the web and http service.
- `portal/help/wwhelp.htm` is the path to the help control file.
- `context=UserConsole` is the context parameter that specifies the document root for the required help collection. This parameter's value is the context for Information Console help, UserConsole, and directs the request to the Information Console help collection. The context value is determined by the Information Console application.
- `topic=Dashboard` is the topic parameter that locates the required help page. This parameter's value is the topic for viewing and navigating the dashboard, Dashboard, which is mapped to an anchor in the `about-ipreports.html` file. The topic value is determined by the Information Console application.

Understanding locale support

Actuate provides help in US English. The documentation installer places this help in `<context root>\help`. The installer creates directories for all available locales within `<context root>\help`. The locale directory names are the locale code of the form `<ll_cc>` where `ll` is a language code and `cc` is a country code. The directory names are all in lowercase letters. Each locale directory contains a `wwhelp.htm` file and directories for each help collection listed in Table 10-2, as shown in Figure 10-5 for the `ac_is` locale.

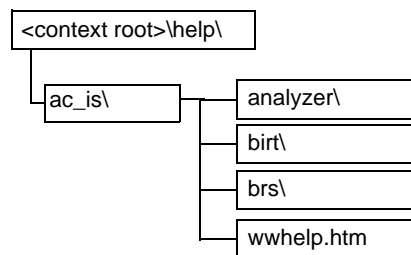


Figure 10-5 `ac_is` locale directory structure

The `wwhelp.htm` files in each locale directory and its collection directories redirect to the files directly in `<context_root>\help`. To support localized online help, place localized files in the appropriate locale directory and modify the `wwhelp.htm` files to not redirect to `<context_root>\help`.

Using a custom help location

Any help system hosted by a web server can provide online help for an Information Console system. To make an external help system available to the Information Console application, the `wwhelp.html` file redirects help requests to that external system. Any specific help target can link to any specific page.

To redirect help requests from Information Console to an alternate URL, edit or replace the `wwhelp.html` file in `<context root>\help`. You can further specify different targets using the context and topic parameters in the URLs emitted by Information Console in help requests.

Customizing the help location with `wwhelp.htm`

Use the following procedure to create a `wwhelp.htm` file that redirects Information Console context-sensitive help requests to another URL.

- 1 In a text editor, open a new document.
- 2 Write the required pieces of an HTML file, as shown in the following code:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<script type="text/javascript" language="JavaScript1.2">
  <!--
  ...
  // -->
  </script>
</head>
<body>
</body>
</html>
```

- 3 Within the script block, write the JavaScript method `GetParameter` to capture URL parameters, as shown in the following code:

```
// get parameters from the URL
//
method GetParameter( name )
{
  var regexS = "[\\?&]+" + name + "=[^&#]*";
  var regex = new RegExp( regexS );
  var results = regex.exec( window.location.href );
  if( results == null )
    return "";
  else
    return results[1];
}
```

4 As shown in the following code, create a method to do the following tasks:

- Operate the page.
- Use `GetParameter` to obtain the topic and context from the URL.
- Open a URL based upon the topic and context.

```
method LaunchHelp()  
{  
  // Get URL parameters  
  var context = GetParameter( 'context' );  
  var topic = GetParameter( 'topic' );  
  
  var baseURL = "http://myhelpserver/viewer/wwhelp.htm";  
  
  // Begin flow control using context  
  switch (context)  
  {  
    // map the "BIRTIV" context to an outside URL  
    case "BIRTIV" :  
      self.location = baseURL + "?single=true&context=" +  
context + "&topic=" + topic ;  
      break;  
  
    // map the "UserConsole" context to an outside URL  
    case "userconsole" :  
      baseURL = "http://myhelpserver/iPortal/wwhelp.htm";  
      self.location = baseURL + "?single=true&context=" +  
context + "&topic=" + topic ;  
      break;  
  
    //the default behavior  
    default :  
      self.location = baseURL ;  
  }  
}
```

The `LaunchHelp()` method gets the context and topic information from the URL with two calls to `GetParameter`. The `baseURL` is set to the myhelpserver application's online help. The flow control switch statements activate specific URLs depending upon the context. Because the myhelpserver application uses the same context and topic variables as standard Information Console help, they are used directly in constructing the URL when activating the `self.location` methods.

5 Replace the `<body>` tag with the body tag in the following line:

```
<body onLoad="LaunchHelp();">
```

The `onLoad` parameter activates `LaunchHelp()` when the page loads.

6 Save the file as `wwhelp.htm` in the `<context root>\help` directory.

- 7 Test the results by opening Information Console and selecting a help link. The resulting page is from the custom application. For example, the help link on the login page pictured in Figure 10-4 would link to http://myhelpserver/portal/help/wwhelp.htm?single=true&context=UserConsole&topic=Login_MyDoc_Enterprise.

Creating a localized help collection

Actuate Information Console supports localizing help collections by placing localized help files into the help directory for the appropriate locale. The `<context_root>\help` directory contains several locale-specific help directories. For example, the United States English help subdirectory is `<context_root>\help\en_us`. Other help locale directories can be populated with localized help to provide help for customers in other locales and in other languages. In order to maintain proper help navigation and context-sensitive help links, localized help pages must have the same name as the help pages provided by Actuate.

How to create a localized help collection

Use the following steps to create a localized online help collection for Information Console that maintains context-sensitive help requests and help navigation.

- 1 Copy all of the non-locale-specific directories from `<context_root>\help` into the appropriate locale-specific directory. For example, for the Italian locale, copy the files into `<context_root>\help\it_it`.
- 2 Create localized versions of existing help files in a separate directory.
- 3 In the locale-specific directory, copy the localized versions of the help files over the English files of the same name. The localized help can be accessed using the following URL:

```
http://localhost:8700/portal/help/<locale-specific directory>/wwhelp.htm
```

For example, for the Italian locale-specific help, use the following URL:

```
http://localhost:8700/portal/help/it_it/wwhelp.htm
```

- 4 Test the results by opening Information Console, selecting the new locale on the login page, and selecting a help link. The resulting page is from the custom application. For example, the help link on the login page shown in Figure 10-4 would link to http://localhost:8700/portal/help/it_it/wwhelp/wwhimpl/common/html/wwhelp.htm#href=using-ip/iPusing-intro.2.06.html#214106&single=true.

How to make locale-specific online help the default help

Use the following procedure to make a locale-specific help collection the default help for Information Console.

- 1 Open `wwhelp.htm` in the `<context root>\help` directory in a text editor. Find the following line:

```
setTimeout("location.replace(\"/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

Add the locale-specific directory to the URL string, as shown in the following code:

```
setTimeout("location.replace(\"/<locale-specific directory>/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

For example, to set the Italian locale as the default locale for context-sensitive help, change the line to the following one:

```
setTimeout("location.replace(\"/it_it/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

- 2 Save and close `wwhelp.htm`.
- 3 Copy all of the non-locale-specific directories from `<context_root>\help` into each English locale-specific directory - `en_au`, `en_bz`, `en_ca`, `en_gb`, `en_ie`, `en_nz`, `en_us`, and `en_za`. For example, for US English, copy the files into `<context_root>\help\en_us`.
- 4 In each English locale-specific directory, open `wwhelp.htm` in a text editor. Find the following line:

```
setTimeout("location.replace(\"../wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\")
```

Add the locale-specific directory to the URL string, as shown in the following code:

```
setTimeout("location.replace(\"/<locale-specific directory>/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

For example, to set US English help to the `en_us` locale for context-sensitive help, change the line to the following one:

```
setTimeout("location.replace(\"/en_us/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

- 5 Test the results by opening Information Console and selecting a help link. The resulting page is from the custom application. For example, the help link on the login page shown in Figure 10-4 would link to `http://localhost:8700/portal/help/it_it/wwhelp/wwhimpl/common/html/wwhelp.htm#href=using-ip/iPusing-intro.2.06.html#214106&single=true`.

Then, test an English locale by selecting an English locale on the login page and then selecting a help link. The resulting page is from the English locale help. For example, the help link on the login page shown in Figure 10-4 would link to `http://localhost:8700/portal/help/en_us/wwhelp/wwhimpl`

/common/html/wwhelp.htm#href=using-ip/iPusing-intro.2.06.html#214106
&single=true for the US English locale.

Customizing icons, links, and the company logo

The online help pages organize navigation and content into frames. To change the fonts, colors, and icons of customized help, change each frame's content or style file individually.

Changing the corporate logo

Each content page contains a small logo in the footer, as shown in Figure 10-6. The image tag in the content page displays the corporate logo in the content frame. To change this logo, change the image tag on every content page.

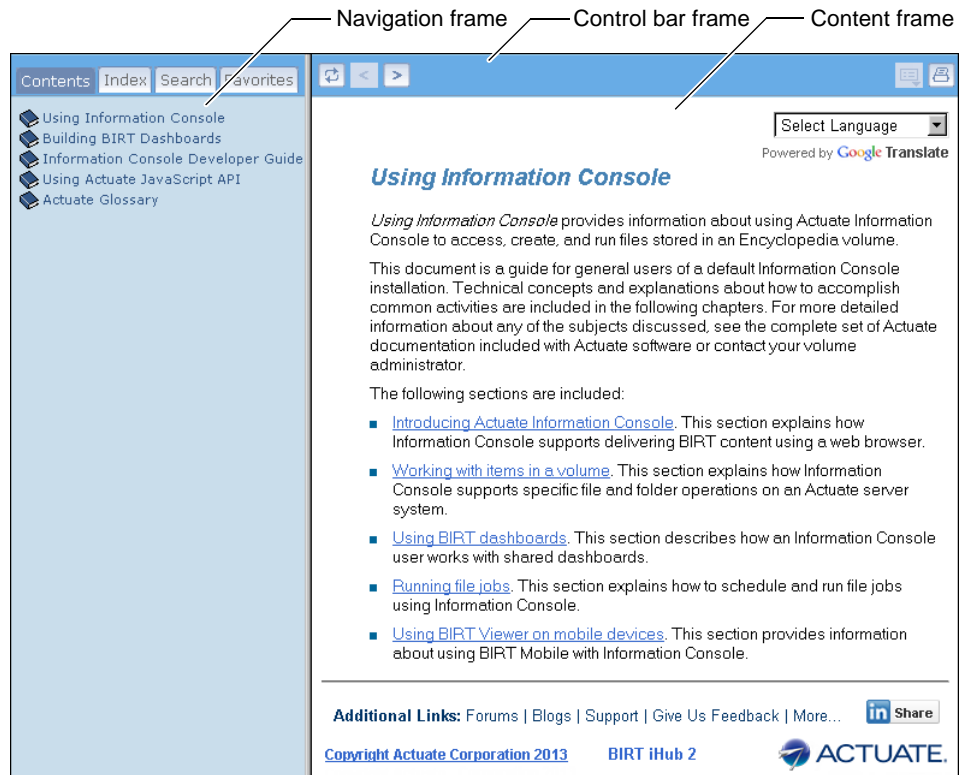


Figure 10-6 Help frames

The footers in the content pages display the Actuate corporate logo by default. To change the corporate logo displayed on a content page, alter the HTML markup

to use a different logo. Actuate uses the corporate logo as a link to the Actuate corporate web site. Change this link so that the image is a link to your corporate web site.

How to change the corporate logo on a help content page

Use the following procedure to alter the corporate logo and corporate web application link in a content page.

- 1 Copy your corporate logo image file into the <document root>\images directory for the help topic content you wish to change. For example, to change the logo in the “Using Information Console” help topic, the document root is the <context root>\help\using-ip directory.
- 2 In a text editor, open the first content page file in the document root. For example, the first content page in the “Using Information Console” documentation is iPusing-intro.2.01.html.
- 3 Locate the following block of code:

```
<table cols="3" summary="" width="100%">
<tr>
<td class="WebWorks_Company_Name_Bottom" align="left"
    width="30%">
<a href="copyright.htm">Copyright Actuate Corporation 2013</a>
</td>
<td style="color: #3165CE;font-family:sans-serif;font-
    weight:bold;font-size:small;" align="center" width="40%">
    BIRT iHub 2
</td>
<td class="WebWorks_Company_Logo_Bottom" align="right"
    width="30%">
<a href="http://www.actuate.com" target="_blank"
    align="right"></a>
</td>
</tr>
</table>
```

- 4 Change <http://www.actuate.com> to the address of your corporate web site.
- 5 Change `logo-small-blue.png` to the name of your corporate logo image file.
- 6 Save and close the content file.
- 7 Repeat steps 2 through 6 for each content file you need to change.

Changing the additional links footer in help content pages

The footers in the content pages also display a series of additional links by default. Actuate uses these additional links to jump to locations in its corporate

web site. To change the links displayed on a content page, alter the HTML markup to use different links.

How to change the additional links footer on a help content page

Use the following procedure to alter the additional links footer in a content page.

- 1 In a text editor, open the first content page file in the document root. For example, the first content page in the "Using Information Console" documentation is `iPusing-intro.2.01.html`.

- 2 Locate the following block of code:

```
<table cols="2" summary="" width="100%">
<td width="80%" align="left">
<table cols="1" summary="" align="left">
<td align="right">
<td style="color: #003366;font-family:sans-serif;font-
weight:bold;font-size:small;" align="right">Additional
Links:</td>
<td class="td-link" align="right">
<A HREF="http://www.actuate.com/actuate11/forums"
target="_blank">Forums |</A>
</td>
<td class="td-link" align="right">
<A HREF="http://www.actuate.com/actuate11/blog"
target="_blank">Blogs |</A>
</td>
<td class="td-link" align="right">
<A HREF="http://www.actuate.com/actuate11/esupport"
target="_blank">Support |</A>
</td>
<td class="td-link" align="right">
<A HREF="#" onclick="sendEmail();"> Give Us Feedback |</A>
</td>
<td class="td-link" align="right">
<A HREF="http://www.actuate.com/actuate11/resources"
target="_blank"> More...</A>
</td>
</td>
</table>
```

- 3 Change all the links to `http://www.actuate.com/` to addresses for your corporate web site.
- 4 Save and close the content file.
- 5 Repeat steps 2 through 4 for each content file you need to change.

Changing the Google translate element in help content pages

A Google translate element appears in the content pages by default. To change the element displayed on a content page, alter the HTML markup to use different content.

How to change the Google translate element on a help content page

Use the following procedure to alter the additional links footer in a content page.

- 1 In a text editor, open the first content page file in the document root. For example, the first content page in the "Using Information Console" documentation is `iPusing-intro.2.01.html`.
- 2 Locate the following blocks of code:

```
<div id="google_translate_element" style="text-align: right;"></div>
```

and:

```
<script>
function googleTranslateElementInit() {
    new google.translate.TranslateElement({
        pageLanguage: 'en'
    }, 'google_translate_element');
}
</script>
<script src="//translate.google.com/translate_a/
element.js?cb=googleTranslateElementInit"></script>
```

- 3 Delete or replace this code with custom content.
- 4 Save and close the content file.
- 5 Repeat steps 2 through 4 for each content file you need to change.

Changing icons

To change the icons for the controls in the navigation frame and the control bar frame, replace the current image files with different ones. The icon images are located in the `<context root>\help\wwhelp\wwhimpl\common\images` directory. Replacing these image files changes the icons used for the control bar and navigation frames. Table 10-5 lists and describes the image files for the icons.

Table 10-5 Help icon image files



















Image	File name	Purpose	Location
	bkmrk.gif	Bookmark the current page.	The control bar frame

Table 10-5 Help icon image files

Image	File name	Purpose	Location
	bkmrkx.gif	The bookmark method is not available.	The control bar frame
	doc.gif	Open a single file in the table of contents.	The navigation frame
	email.gif	E-mail a link to the current page.	The control bar frame
	emailx.gif	E-mailing a link is not available.	The control bar frame
	fc.gif	Expand a help topic or sub-topic in the table of contents.	The navigation frame
	fo.gif	Collapse a help topic in the table of contents.	The navigation frame
	shownav.gif	Open the control frame.	The control bar frame
	next.gif	Go to the next page.	The control bar frame
	nextx.gif	There is no next page available.	The control bar frame
	prev.gif	Go to the previous page.	The control bar frame
	prevx.gif	There is no previous page available.	The control bar frame
	print.gif	Print the current page.	The control bar frame
	printx.gif	Printing is not available for this page.	The control bar frame
	related.gif	View related topics.	The control bar frame
	relatedx.gif	The related topics method is not available.	The control bar frame
	sync.gif	Synchronize the frames so that the control frame matches the content frame.	The control bar frame
	syncx.gif	Synchronizing frames is not available.	The control bar frame

Changing the browser window title

To change the title displayed in the browser's title bar when viewing online help, alter the title.js file for each document root. The browser's title bar appears as shown in Figure 10-7.

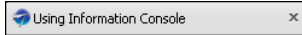


Figure 10-7 The browser's title bar

How to change the text displayed in the browser's title bar

Use the following procedure to change the text displayed in the browser's title bar when you access help.

- 1 Navigate to the <document root>\wwhdata\common directory for the help topic you want to customize. For example, to change the text displayed in the browser title bar when you open the "Using Information Console" help topic, the <document root> is the <context root>\using-ip directory.
- 2 In a text editor, open title.js.
- 3 Locate the line in the code that uses the return method. For the "Using Information Console" help topic, it is the following line:

```
return "Using Information Console";
```
- 4 Change the quoted text value to the text you need to display in the browser's title bar.
- 5 Save and close title.js.

Changing help content

Every piece of content in the Actuate Information Console help system is customizable. The possible content changes fall into the following general categories:

- Changing existing help content
- Adding or removing help topics
- Adding and removing content files
- Changing the table of contents
- Changing the index

Changing existing help content

You can modify any of the existing HTML pages of the Information Console help for any help topic to change the information they contain. These HTML files

contain specific `<a>` tags used for internal navigation and context-sensitive help. In general these tags must remain unchanged to maintain context-sensitive help and internal navigation functionality. Table 10-6 lists the tags and their use.

Table 10-6 Help content reserved tags

Tag examples	Purpose
<code></code>	An anchor for a specific place in a file. This tag is used by internal links and context-sensitive links.
<code><a href="javascript:WWHClickedPopup('UserConsole', 'iPusing-intro.2.02.html#150156', ");</code>	Internal link. This tag is an internal link to an anchor. In this example: <ul style="list-style-type: none">■ UserConsole is the context, a reserved help topic label.■ 'iPusing-intro.2.02.html' is the file that the link opens.■ #150156 is the text of the anchor tag that the link accesses.

How to modify the content of existing pages

Use the following procedure to change the help content.

- 1 Navigate to the document root directory for the help topic you want to change. For example, to change the content of a page in the “Using Information Console” help topic, the document root is the `<context root>\using-ip` directory.
- 2 In a text editor, open the content page you need to change. For example, to change the content of the “Working with items in a volume” page, open the `iPusing-reports.3.01.html` file.
- 3 Modify the text, being careful not remove any `<a>` tags that provide internal links and context-sensitive links.
- 4 Save and close the content file.

Adding or removing help topics

To add or remove help topics from the application help, you delete or create the document root for that help topic. To prevent the navigation pane controls from generating erroneous links to that help topic, you must also alter the help book list, `books.js`, located in the `<context root>\help\wwhelp\wwhimpl\common\private` directory. The `books.js` file also controls the order in which the help topics appear in the table of contents.

How to remove a help topic from the Information Console help system

The following steps remove a topic from the Information Console help system.

- 1 Navigate to the <context root>\help\wwhelp\wwhimpl\common\private\ directory.
- 2 In a text editor, open the books.js file.
- 3 Find the following code:

```
function WWHBookGroups_Books (ParamTop)
{
    ParamTop.fAddDirectory("using-ip", null, null, null, null);
    ParamTop.fAddDirectory("dashboards", null, null, null, null);
    ParamTop.fAddDirectory("customizing-ip", null, null, null,
    null);
    ParamTop.fAddDirectory("javascriptapi", null, null, null,
    null);
    ParamTop.fAddDirectory("glossary", null, null, null, null);
}
```

- 4 Delete the line that adds the directory for the topic that you need to remove.
- 5 Save and close the books.js file.
- 6 In the file system, delete the document root for the topic that you removed in step 4.

Adding and removing content files

Individual content files are added or removed from the document root for each top-level help topic. To make the content file available for linking and viewing from the help system, you must also alter the file list, files.js, located at <document root>\wwhdata\common. The files.js file also controls the order of the files in the array for reference by other files. For example, the content of files.js for the using-ip document root looks like the following code:

```
function WWHBookData_Files (P)
{
    P.fA("Using Information Console", "about-ipreports.html");
    P.fA("Introducing Actuate Information Console", "iPusing-
    intro.2.01.html");
    P.fA("Delivering BIRT content", "iPusing-intro.2.02.html");
    P.fA("About Information Console", "iPusing-intro.2.03.html");
    ...}
```

This code establishes the following structure:

- Each file, about-ipreports.html, iPusing-intro.2.01.html, iPusing-intro.2.02.html, and iPusing-intro.2.03.html, is available for linking and display by Information Console help.
- The first file in the array is about-ipreports.html, which is referenced by the array number 0. The second file in the array is iPusing-intro.2.01.html and is referenced by the array number 1 and so on.

The order of the files in the array always begins with and proceeds from 0. The file array is an internal mechanism that supports referencing these files by number within the help topic.

How to add a content file to the Information Console help system

Use the following procedure to add a content file to the Information Console help system.

- 1 Copy your content file into the document root directory for the help topic you need to enhance. For example, to add a new file to the “Using Information Console” help topic, the document root directory is <context root>\using-ip.
- 2 Navigate to the <document root>\wwhdata\common directory.
- 3 In a text editor, open the files.js file.
- 4 Find the following code:

```
function WWHBookData_Files(P)
{
  P.fA("Using Information Console", "about-ipreports.html");
  P.fA("Introducing Actuate Information Console", "iPusing-
    intro.2.01.html");
  P.fA("Delivering BIRT content", "iPusing-intro.2.02.html");
}
```

- 5 Add a P.fA(...); entry for the file to add, placing it where you need it to appear in the file array relative to the other entries.

P.fA(...); requires two parameters. The first is a string that describes the file. The second is the name of the file. Both parameter values must individually be within quotation marks and separated by a comma.

- 6 Change the parameter values for the other P.fA(...) calls as needed.
- 7 Save and close files.js.

Changing the table of contents

Help topics are established in the table of contents by the title.js file in the <document root>\wwhdata\js\ directory for each help topic. For example, the title.js file for the using-ip document root looks like the following code:

```
method WWHBookData_Title()
{
  return "Using Information Console";
}
```

This code sets the table of contents text for this help topic to “Using Information Console.” Figure 10-8 shows the hierarchy produced by the code above.

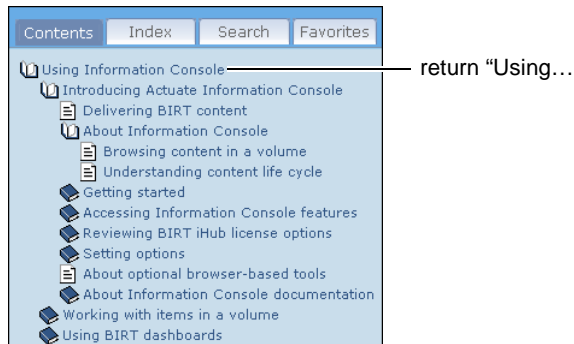


Figure 10-8 The help topic entry in the table of contents

The table of contents displays nested help topics as listed in the `toc.js` file located in the `<document root>\wwhdata\js` directory. The `toc.js` file also controls the following items:

- The table of contents hierarchy
- The text that appears in the table of contents
- The file that opens when a user selects a table of contents entry

For example, the following code is part of the table of contents for the “Using Information Console” chapter in the `toc.js` file for the `using-ip` document root:

```
var A=P.fN("Introducing Actuate Information Console","1");
var B=A.fN("Delivering BIRT content","2");
B=A.fN("About Information Console","3");
var C=B.fN("Browsing content in a volume","4");
C=B.fN("Understanding content life cycle","5");
B=A.fN("Getting started","6");
C=B.fN("How to log in to a volume using Information Console",
    "6#214106");
```

This code establishes the following structure:

- The top-level entry, A, is file "1". File 1 is in position 1 of the internal file array established by `files.js`. For example, in the `using-ip` document root, this file is `iPusing-intro.2.01.html`.
- Entries are created to reside in the next level under the top-level entry using the variable B. Entries in the third level of the table of contents are created using the variable C, and in the fourth level using the variable D. An entry links to a file or an anchor within a file referenced by the internal file array number. For example, "6#214106" links to the `` anchor in file "6" of the file array, `iPusing-intro.2.06.html`.

- The text that appears in the table of contents for each entry is explicitly defined. For example, the text for the top-level entry is “Using Information Console”.

Figure 10-9 shows the hierarchy produced by this code.

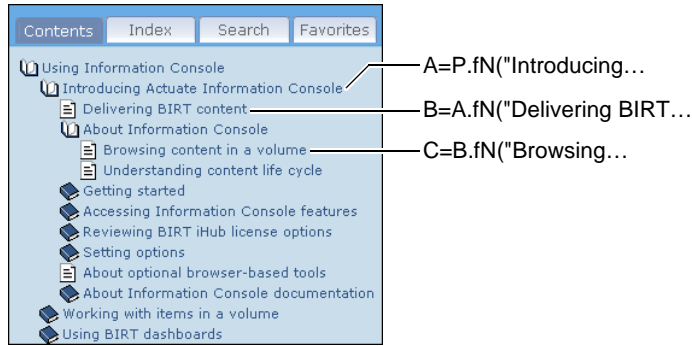


Figure 10-9 The table of contents hierarchy for using-ip

How to add a content file link to the table of contents hierarchy

Use the following procedure to add a content file link to the table of contents hierarchy for the Information Console help system.

- 1 If you are linking to an anchor, navigate to the document root directory. Open the content file that contains the anchor to which the table of contents will link. Determine the value of the name attribute for the anchor. Then, close the content file without saving it.
- 2 Navigate to the <document root>\wwhdata\common directory.
- 3 In a text editor, open the files.js file and determine the internal file array number for the content file, either that you opened in step 1 or that you are linking to directly. Close files.js without saving it.
- 4 Navigate to the <document root>\wwhdata\js directory.
- 5 In a text editor, open the toc.js file.
- 6 Add an entry to toc.js for the table of contents entry using the following format:

```
var D=C.fN("Setting an e-mail address","15#228413");
```

- var is a keyword that must precede the entry if D has not been defined as a variable in this file prior to this line. Do not use var if D has already been defined.
- D is the table of contents hierarchy level of the new table of contents entry.
- C is the table of contents hierarchy level above the level of the new table of contents entry.

- "Setting an e-mail address" is the string to display in the table of contents for this entry.
- 15 is the array number of the target file established in step 3.
- #228413 is a number sign (#) followed by the value of the name attribute for the anchor established in step 1, if it is applicable. To link to the head of a file, do not append an anchor string to the array number of the target file.

7 Save and close toc.js.

Changing the index

The index displays keywords for help topics from individual content files. The index.js file located in the <document root>\wwhdata\js directory contains the index entries. The help system merges the index.js files for every book in a collection. The index.js file controls the following items:

- The index hierarchy
- The text that makes up the index entries
- The content to which the index entries link

For example, in the using-ip document root, the index entry for "changing", starting at the letter C, looks like the following code:

```
A=P.fA("C",null,null,"002");
B=A.fA("calendar gadgets",new Array("91#758834"));
B=A.fA("calendars",new Array("87#808008"));
B=A.fA("cancelling scheduled jobs",new Array("130"));
B=A.fA("Categories view",new
    Array("8#249139","8#214378","42#405845"));
B=A.fA("changing");
C=B.fA("dashboards",new
    Array("71#783190","72#786017","83#774197","88#775563"));
C=B.fA("data",new Array("94#752260"));
C=B.fA("data sources",new Array("5#191220"));
```

This code establishes the following structure:

- The top-level entry, A=P.fA, is the label "C". This entry links to the "002" frame, which is the navigation frame.
- The first entry below "C" is the "calendar gadgets" entry. This entry is one level down in the hierarchy, B=A.fA, of the index for "C". This entry has one link to an anchor in file "91".
- The entry for "changing" is merely a label and does not link to anything.
- On the next level down in the hierarchy, C=B.fA, has many entries, one for each of the sub-topics of changing. Each entry has a label and an array of links to topics that the user can choose.

Figure 10-10 shows the hierarchy produced by this code.

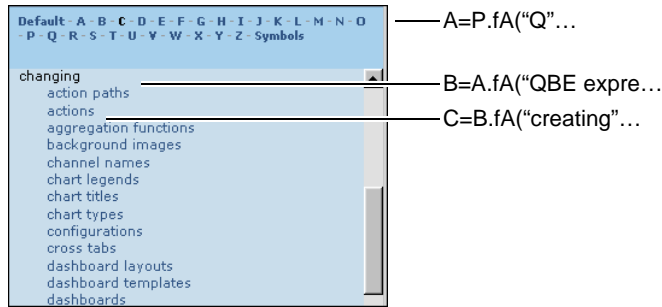


Figure 10-10 The index hierarchy for using-ip

How to add a marker link to the index hierarchy

Use the following procedure to add a marker link to the index hierarchy of the Information Console help system.

- 1 Navigate to the document root directory. Open the content file that contains the anchor to which the index entry will link. Determine the value of the name attribute for the anchor. Then, close the content file without saving it.
- 2 Navigate to the <document root>\wwhdata\common directory.
- 3 In a text editor, open the files.js file and determine the internal file array number for the content file that you opened in step 1. Close files.js without saving it.
- 4 Navigate to the <document root>\wwhdata\js directory.
- 5 In a text editor, open the index.js file.
- 6 Add an entry to index.js for the index entry and anchor link using the following format:

```
var B=A.fA("access restrictions",new Array("43#407680"));
```

- var is a keyword that must precede the entry if B has not been defined as a variable in this file prior to this line. Do not use var if B has already been defined.
- B is the index hierarchy level of the new index entry.
- A is the index hierarchy level above the level of the new index entry.
- "access restrictions" is the string to display in the index for this entry.
- 43 is the array number of the target file established in step 3.
- #407680 is a number sign (#) followed by the value of the name attribute for the anchor established in step 1.

To link the index entry to more than one marker, add each marker link to the list within the new Array parameters. Enclose each anchor reference in quotation marks. Delimit the anchor references with commas, shown in the following example:

```
var B=A.fA("access restrictions",new  
    Array("43#407680", "46#408024"));
```

- 7** Save and close index.js.

Index

Symbols

- _ (underscore) character 13
- ; (semicolon) character 13
- : (colon) character 12
- ! (exclamation point) character 13
- ? (question mark) character 13
- . (period) character 13
- , (comma) character 13
- " (double quotation mark) character 13
- { } (curly brace) characters 129
- @ (at-sign) character 12
- * (asterisk) character 12, 123
- / (forward slash) character 13
- \ (backslash) character 12
- & (ampersand) character 12
- # (number sign) character 13
- % (percent) character 13
- + (plus sign) character 13
- < (less than) character 13
- = (equals sign) character 13
- > (greater than) character 13
- \$ (dollar sign) character 13

A

- about page 82, 92, 95
- accessing
 - application servers 44
 - cascading style sheets 44, 46
 - channels 10, 74
 - dashboards 10
 - Encyclopedia volumes 79, 168
 - help content pages 185, 186
 - home page 107
 - Information Console functionality 11, 42, 64, 72, 136, 158
 - Interactive Viewer 17
 - JavaScript files 44, 46
 - JSPs 24
 - report files 159
 - repository items 10, 74, 112
 - resource bundles 144, 149
 - resources 18, 46, 70
 - session-specific information 43
 - skin manager 51
 - tag libraries 44
 - templates 45
 - web applications 27, 108
- accessToGrant parameter 127
- AcChannelFilter variable 97
- AcFilesFoldersFilter variable 113
- AcFilesFoldersTypeFilter variable 113
- AcGetFileDetailsAction class 159
- AcGetFolderItemsAction bean 42
- AcGetJobDetailsAction class 160
- AcServlet class 136
- actabpanel tag library 142, 143
- Action class 42, 158
- action forms 158, 160
- action forms classes. *See* forms package
- action parameter 121
- action path names 48
- action paths 22, 42, 43, 48
 - See also* URIs
- action tag 42
- ActionForm class 158
- actions 17, 22, 74, 82, 85
- actionServlet component 22
- activePortal directory 25
- activity logs 67, 68
- acweb.war 6
- AddFile subfeature 17, 75
- adding
 - action paths 42, 43, 48
 - background images 59
 - context roots 27–29
 - custom JSPs 24, 45
 - custom security adapters 172, 173–174, 175
 - features 18, 20
 - folders 17, 98, 159
 - functionality levels 14, 15, 16, 20
 - help content files 200, 201
 - help topics 199
 - hyperlinks 160
 - locales 75

- adding (*continued*)
 - server profiles 32
 - skins 51, 52, 53, 167
 - time zones 76
 - upload security adapters 178, 179, 179–181
 - URI parameters 12, 13
 - volume profiles 70–72
 - web pages 10, 43, 44, 48
- addressing e-mail 117
- Administrator functionality level 14, 15, 73
- Advanced functionality level 14, 15, 73
- AdvancedData subfeature 17, 75
- ageDays parameter
 - execute report page 104
 - submit job page 127
- ageHours parameter
 - execute report page 104
 - submit job page 127
- aging intervals 104, 127
 - See also* archive policies
- allscripts.js 132
- analytics gadgets 65
- Apache Jakarta Struts. *See* Jakarta Struts
- Apache Tomcat engine 27
- Apache Tomcat service 29
- application context roots 11, 23, 43
- application servers 5, 7, 27
 - See also* servers
- applications
 - accessing 27, 108
 - building user interface for 4, 45, 49, 142
 - changing 6, 29, 41, 49
 - configuring 39, 64
 - creating context root for 27–29
 - creating page-specific content for 48
 - customizing pages for. *See* web pages
 - deploying 170
 - designing custom reporting 8, 22, 29, 39, 49
 - determining state of 44
 - getting session information for 43
 - grouping 26
 - linking help files to 186, 187, 194, 199, 200
 - previewing skins for 51, 52
 - setting default locale for 66, 76
 - setting default time zone for 66
 - setting global styles for 49–59
 - testing 27
 - translating. *See* locales
- applyFilter parameter 110, 113
- archive policies 104, 127
- archiveBeforeDelete parameter
 - execute report page 104
 - submit job page 127
- archivePolicy parameter
 - execute report page 104
 - submit job page 127
- archiving report objects 104, 127
- array elements 132
- array.js 132
- arrays 132
- ASCII formats 77
- attachments 17
- authenticate method 171, 173, 175
- authentication
 - accessing applications and 170
 - accessing corporate networks and 170
 - customizing 170, 173
 - issuing URIs and 84
 - logging in to Information Console and 171, 172
- authentication IDs 44, 84, 147, 163, 166
- authentication information 7
- authexpired action 86
- authID variable 148
- AUTO_SAVE_DASHBOARD_DELAY
 - parameter 79
- autoarchive policies 104, 127
- autoarchiving. *See* archiving

B

- background images 59
- backward compatibility 24
- banner
 - adding links to 72
 - changing welcome text in 41
 - displaying 95, 114
 - hiding features in 16
 - replacing images in 59
 - specifying functionality levels and 15
- banner elements 109, 114
- banner labels 58
- banner page 82, 92, 95

- banner styles 58
- BaseActionForm class 160
- Basic functionality level 14, 15, 73
- beans 43, 45, 57, 158, 165
- beans package 158, 162
- binary files 136
- BIRT 360 web resources 65
- BIRT Data Analyzer 80, 186
- BIRT iHub 28
- BIRT iHub System. *See* iHub System
- BIRT Interactive Viewer. *See* Interactive Viewer
- BIRT report documents 137
 - See also* BIRT reports
- BIRT reports 80, 130
 - See also* reports
- BIRT Studio 49, 64, 80, 186
- BIRT Studio link 107
- BIRT Studio pages 58
- BIRT Viewer 64, 80, 130
- BIRT Viewer help collections 186
- BIRT Viewer servlet 130
- BIRT_RENDER_FORMAT_EMITTER_ID_MAPPING parameter 65
- BIRT360PLUS_URL parameter 65
- body element 41
- Bookmark icon 196
- bookmark parameter 138
- branding 41, 59
- breadcrumbs 47, 107
- browse file page 82, 92, 96
- browse page. *See* browse file page
- browsefile action 86, 87, 159
- BrowseFileActionForm class 159
- browseportletfile action 87
- browsers. *See* web browsers
- browsertype.js 132
- browsing 96, 159
- bundle tag 143, 144
- BundleTag class 144
- bundling resource files 144, 149
- buttons 49

C

- CACHE_CONTROL parameter 65
- caching web pages 41, 65, 79

- calendar 132
- calendar page 82, 92, 96
- calendarlayer.js 132
- canceljob action 86, 87
- canceljob page. *See* request drop page
- cancelreport action 87
- cascading style sheets
 - accessing 44, 46
 - changing styles in 57
 - customizing web pages and 40, 59
 - linking to JSPs 56, 57
 - specifying color settings in 57
 - updating changes to 42
 - viewing changes to 58
- case sensitivity 22, 82, 127, 136, 142
- cbFail parameter 110
- cbSuccess parameter 110
- ChangeResponseLocale message 144
- changing
 - action paths 43
 - actions 74
 - background images 60
 - company logos 193–196
 - configurations 29, 30
 - default servers 32
 - Encyclopedia volumes 32
 - error messages 34
 - file names 48
 - font styles 57
 - functionality levels 20
 - help content 198, 199
 - help indexes 204
 - icons 74, 196
 - images 41, 52, 59–60
 - label key values 18
 - landing pages 40–41
 - link icons 19
 - link targets 18
 - locales 30, 118
 - passwords 118
 - reporting applications 6, 29, 41, 49
 - servlets 136
 - side menu 74
 - style definitions 56, 57
 - templates 45, 46
 - time zones 30
 - user interface elements 60

- changing (*continued*)
 - web browser titles 198
 - web pages 46, 48
- channel classes 158
- channel contents list page 94, 112
- channel icons 117
- channel parameter 112
- channelIcons parameter 117
- channelID parameter 99
- ChannelListActionForm class 158
- channelName parameter 99, 102, 110
- channels
 - accessing 10, 74
 - displaying 111, 117, 158
 - filtering 97
 - removing notifications from 99
 - sending notifications to 110
 - setting properties for 96
 - subscribing to 17, 111, 159
 - unsubscribing from 111
 - viewing contents 10, 112
- Channels attribute (features) 16, 19, 74
- Channels link 19
- channels list page 94, 111
- channels page 82, 92, 96
- channels parameter 117
- character encoding 12, 14, 77, 132
- character sets 14
- character substitution 12
- check boxes 113
- class reference (JavaBeans) 158, 162
- classes 40, 173, 179
- Classic skins 49
- clearFilter parameter 110
- cloning skins 51, 52, 53
- cluster nodes 79
- clusters 4, 5, 6, 7
- code 44
- color names 52, 57
- color palettes 52
- colors 49, 55, 57
- comments 41
- common tag library 142, 143
- company logos 59, 193–196
- compiling JSPs 22
- completed jobs page 110
 - See also* completed request page
- completed request page 82, 93, 97
- completion notices 117
- configuration files 30, 64
- configuration parameters
 - BIRT Data Analyzer 80
 - changing 30
 - iHub connections 79
 - Information Console 30, 64
- configurations
 - accessing Information Console
 - functionality and 42, 64, 72
 - adding web pages and 43
 - changing messages and 34, 76
 - connecting to iHub and 70, 79
 - creating custom applications and 29–39, 64
 - customizing context root and 27
 - defining features and 16, 18, 74
 - defining functionality levels and 14, 16, 18, 20, 72–75
 - defining subfeatures and 17, 75
 - deploying an upload security adapter and 180
 - deploying IPSE applications and 175
 - disabling load balancing and 7
 - generating locale-specific sites and 75, 76
 - initiating actions and 85
 - invoking servlets and 136
 - redirecting web pages and 9
 - renaming files and 48
 - running BIRT reports and 80
 - running multiple applications and 7
 - setting default volume and server 32
 - setting up firewalls and 8, 9, 171
 - specifying time zones and 76
 - updating changes to 29
- confirmation messages 45
- confirmKey parameter 117
- connection information 64
- connection parameters 22
- connection pools 79
- CONNECTION_TIMEOUT parameter 66
- connections
 - accessing Encyclopedia and 22, 85, 122, 177
 - accessing private networks and 171
 - dropping 66

- establishing iHub 79, 147, 171
 - protecting data and 170–171
 - setting maximum number of 79
 - timing out 68
- Content class 145
- content element 48
- content tag 143, 145
- context menus 59
- context roots 11, 23, 27, 43
- context-sensitive help 132, 187, 190
- converter.js 132
- COOKIE_DOMAIN parameter 66
- COOKIE_ENABLED parameter 66
- COOKIE_SECURE parameter 66
- cookie.js 132
- cookies 66, 84, 113, 132
- copyFileFolder tag 143, 146
- CopyFileFolderTag class 146
- copying
 - folders 146
 - image files 60
 - report files 146
 - skins 51, 52, 53
- corporate logos. *See* company logos
- country codes 76, 77
- create folder page 83, 93, 98
- createfolder action 87, 159
- CreateFolder subfeature 17, 75
- CreateFolderActionForm class 159
- creating
 - action paths 42, 43, 48
 - context roots 27–29
 - custom JSPs 24, 45
 - custom security adapters 172, 173–174, 175
 - features 18, 20
 - folders 17, 98, 159
 - functionality levels 14, 15, 16, 20
 - help files 189, 191
 - help indexes 205
 - hyperlinks 160
 - skins 51, 52, 53, 167
 - upload security adapters 178, 179, 179–181
 - URIs 11, 12
 - user interfaces 4, 45, 49, 142
 - WAR files 5, 6
 - web applications 8, 22, 27, 39–48
 - web pages 10, 44, 48
- credentials. *See* login credentials
- cross tabs 80
- CSS files 40, 44, 56, 58
 - See also* cascading style sheets
- custom security adapters 173
- custom tags. *See* JSP custom tag reference
- Customization attribute (features) 16, 74
- Customization link 15, 19
- customize action 87
- customizing
 - background images 60
 - features 18–19
 - file verification 179
 - functionality levels 16–18
 - Information Console 5, 6, 30, 49
 - JSPs 24, 44, 45, 46
 - landing page 40–41
 - messages 34–39, 76
 - online help 184–193, 198
 - reporting applications 8, 22, 27, 39–48
 - security adapters 172–174, 175
 - skins 49, 51, 52, 53, 74, 161
 - upload security adapters 178–180
 - user authentication 170, 173
 - user logins 170
 - web pages 39, 46, 48

D

- da action 86
- daemonURL parameter 115
- dashboard action 86
- dashboard files 45
- dashboard page 83, 93
- dashboard templates 44, 45, 46
- DASHBOARD_SHARED_RESOURCES
 - parameter 79
- DashboardBusinessUser subfeature 17, 75
- DashboardDeveloper subfeature 17, 75
- dashboards 5, 10, 17, 75, 79, 115
- dashboards servlet 10
- data
 - displaying 22, 48
 - filtering 159, 160, 167
 - protecting corporate 170
 - protecting system. *See* security

- data (*continued*)
 - restricting access to 170
 - synchronizing 17
- Data Analyzer 80, 186
- data filters. *See* filters
- data models 22
- data repositories. *See* Encyclopedia volumes
- database servers 8
- date formats 146
- date patterns 146
- date stamps 106
- dateToDelete parameter
 - execute report page 104
 - submit job page 127
- debug pages 144
- debugging messages 66
- decimal values 57
- default authentication 170
- default banner 95
- default context root 23
- default encoding 14
- default Encyclopedia volume 32, 166
- default error codes 76
- default file names 48
- default functionality levels 72
- default images 59
- default locale 18, 30, 31, 66, 76, 165
- default settings 30, 31, 32
- default skins 49, 51, 52
- default time zone 66
- DEFAULT_LOCALE parameter 30, 66
- DEFAULT_PAGE_BREAK_INTERVAL parameter 66
- DEFAULT_TIMEZONE parameter 30, 66
- delete file status page 83, 93, 98
- delete job page 83, 93, 99
- delete status page 83, 93, 99
- deletefile action 86, 88
- deletefile page. *See* file drop page
- DeleteFile subfeature 17, 75
- deletefilestatuspage. *See* delete file status page
- deletefolder page. *See* folder drop page
- DeleteFolder subfeature 17, 75
- deletejob action 86, 88
- deletejob page. *See* delete job page
- deletejobnotice action 86, 88
- deletejobnotice page. *See* delete status page
- deletejobschedule action 86, 88
- deleting
 - archived reports 105, 129
 - folders 17, 102
 - help content files 200
 - help topics 199
 - jobs 99, 103
 - notifications 99
 - report files 17, 98, 102
 - skins 51
- deploying
 - Information Console 4, 5, 6, 78
 - reporting applications 170
 - reports 5
 - security adapters 174, 175, 180
- designing custom web applications 8, 22, 29, 39, 49
- destination parameter 122
- detail pages 83, 93, 100
- details icon 60
- developers 4
- diagnostic information 120, 122
- dialog boxes 58
- dialogs
 - defining tabs for 150, 153, 155
 - setting orientation of 154
 - setting tab sequence for 151, 152
 - specifying content for 145, 153
- directories 23, 24, 28, 184
- directory names 22
- directory paths. *See* paths
- disk space 69, 122
- display names 76
- displaying
 - banners 95, 114
 - color settings 55
 - completed jobs 97
 - current jobs 123
 - data 22, 48
 - error messages 103, 119
 - failed jobs 110
 - files and folders list 67
 - folders 114, 167
 - help topics 201, 203
 - locales 75
 - login page 12, 114

- pending jobs 125
- report executables 114, 167
- report parameters 120, 128
- reports 10, 11, 24, 69, 80, 132, 137
- repository information 47
- search results 49
- subscribed channels 96, 111, 117, 158
- web pages 42
- DisplayName tag 76
- distributed iHub System. *See* clusters
- do directive 82
- .do file name extensions 22
- do_executereport.jsp 137
- do_update page. *See* options page
- doc format value 138
- docChanFilters parameter 117
- document classes 159
- document files 100, 114, 167
- document root 186
- documentation vii
- documents. *See* reports
- Documents attribute (features) 16, 74
- Documents link 19
- Documents page 47, 159
- doEndTag method 144
- domains 66
- downloadfile action 86
- DownloadFile servlet 137
- DownloadFile subfeature 17, 75
- downloading
 - binary files 136
 - reports 137
- drift.js 132
- drop pages 83, 93, 102
- dynamic data 132

E

- EAR deployments 78
- editTableRow action 88
- E-mail icon 197
- email parameter 117
- e-mail. *See* notifications
- emitters 65
- ENABLE_CLIENT_SIDE_REDIRECT
 - parameter 9, 66
- ENABLE_DEBUG_LOGGING parameter 66

- ENABLE_ERROR_LOGGING parameter 67
- ENABLE_JUL_LOG parameter 67
- encode method 13, 132
- encoder.js 132
- encoding 12, 14, 77, 132
- Encyclopedia volume icons 59
- Encyclopedia volumes
 - See also* repositories
 - accessing 79, 168
 - adding objects to 17
 - connecting to 22, 85
 - creating folders for 98, 159
 - deleting objects in 17, 98, 102
 - downloading from 17, 137
 - getting current 177
 - getting information about 100, 122
 - running Information Console and 4
 - specifying default 32, 166
 - testing connections to 122, 177
 - writing reports to 106, 129
- engines 68
- environment settings 107
- erroportlet action 86
- error action 86
- error codes 37, 76
- error detail page 93, 100
- error levels 37
- error log files 67, 68
- error messages
 - customizing 34, 36–39
 - displaying 103, 119
 - localizing 76–77
- error page 83, 93, 103
- ERROR_LOG_FILE_ROLLOVER
 - parameter 67
- errors 42, 45, 67
 - See also* error messages
- executable files
 - displaying 114, 167
 - generating output for 118, 124, 126
 - selecting 113
 - viewing parameters for 120
- executableName parameter
 - execute report page 104
 - submit job page 127
- execute report page 83, 93, 103

- EXECUTE_DASHBOARD_GADGET_
- GENERATION_WAIT_TIME parameter 67
- EXECUTE_REPORT_WAIT_TIME
- parameter 67, 103
- executedocument action 86, 88
- executereport action 86, 88
- executereport page. *See* execute report page
- executing
 - Java servlets 136
 - jobs 10
 - reports 103, 125
- execution requests. *See* jobs; requests
- expiration intervals 104, 127
 - See also* archive policies
- extended character sets 14
- extensible markup language. *See* XML
- external help systems 189

F

- Factory service 123
- failComp parameter 117
- failed jobs 110
- failEmail parameter 117
- failover 7
- failure notices 117
- feature definitions 16, 18, 74
- feature IDs 18, 74
- feature lists 163, 166, 167
- feature names 18
- Feature tag 74
- FeatureID tag 16, 18, 74
- FeatureOptionsBean class 162
- features 16, 18–20, 73, 162, 167
- file detail page 93, 100
- file drop page 93, 102
- file IDs 137
- file index page 93, 108
- file list page 94, 112
- file lists 67, 113, 159, 166
- file names 19, 22, 48, 74, 120
- file numbers 68
- file paths. *See* paths
- file structures 6
- file type verification 179, 181
- file types 178
- filefoldersprivilege action 88, 159

- FileFoldersPrivilegeActionForm class 159
- fileId parameter 137
- FileListActionForm class 159
- filename parameter 122
- files
 - See also* specific type
 - accessing 10, 74, 112, 159
 - archiving 104, 127
 - assigning privileges to 123
 - assigning to template elements 45
 - changing images and 59, 60
 - converting to ASCII 77
 - copying 146
 - deleting 17, 98, 102
 - displaying 114
 - downloading 137
 - filtering 112
 - generating locale-specific sites and 33
 - generating online help and 185
 - getting information about 100, 159
 - linking to 48
 - locating JSP 42
 - naming 105, 119, 128
 - overwriting 105, 118, 128
 - renaming 48, 60
 - saving 105, 118, 133
 - searching 125
 - sharing 17
 - updating changes to 41
- FILES_DEFAULT_VIEW parameter 67
- filesfolders JSPs 10
- filesfolders tag library 143
- FileUploadActionForm class 161
- filter action forms 159, 160
- filter parameter 110, 113
- filtering
 - channels 97
 - data 159, 160, 167
 - jobs 110
 - report documents 112
- filters 110, 113, 114, 166
- firewalls 8, 9, 170
- flashchartsxml format value 138
- floatingfooter parameter 138
- folder detail page 93, 100
- folder drop page 93, 102
- folder icons 59

- folder IDs 98
- folder index page 93, 108
- folder list page 94, 112
- folder lists 67, 113, 159, 165, 166
- folder names 82, 98, 166
- folder parameter 114, 126
- folders
 - accessing 10, 74, 112
 - archiving contents 104
 - assigning privileges to 123
 - copying 146
 - creating 17, 98, 159
 - deleting 17, 102
 - displaying 114, 167
 - getting home 147, 163, 177
 - linking to 107
 - navigating through 24, 96
 - searching 74, 125
 - specifying type 119, 166
 - viewing information about 100, 114, 159
- folderType parameter 127
- fonts 49, 55, 57
- footers 138
- FORCED_GC_INTERVAL parameter 67
- forceLogin parameter 84, 171
- format parameter (Interactive Viewer) 138
- formatDate tag 143, 146
- FormatDateTag class 147
- formats
 - displaying reports and 138
 - localizing reports and 146
- formatting
 - date values 146
 - web pages 48
- FormFile class 161
- forms. *See* action forms
- forms classes. *See* forms package
- forms package 158
- forward definitions 43, 85
- forward tag 42
- free disk space 122
- from_page_range parameter 139
- from_page_style parameter 139
- functionality levels
 - adding 14, 15, 16, 20
 - associating with users 14, 16

- changing 20
- configuring 72–75
- customizing 16–18
- naming 16, 73
- preserving 20
- specifying features for 16, 18, 19, 73
- specifying subfeatures for 17, 75
- functionality-level.config 74

G

- gadget gallery 79
- GADGET_GENERATION_WAITING_TIME
 - parameter 67
- gadgets 5, 67, 75
- gadgets interface 10
- garbage collection 67
- general options page 83, 93, 106
- GeneralFilterActionForm class 159, 160
- Generate Web Archive option 6
- generating
 - locale-specific sites 33
 - notifications 117
 - output 118
 - WAR files 5, 6
 - web pages 10–11, 22, 48
- getAcLocale method 163
- getAdminRights method 163
- getAuthid method 163
- getContextPath method 43
- getCurrentfolder method 163
- getDefaultServerURL method 163
- getDefaultVolume method 163
- getErrorMessage method 179, 181
- getExtendedCredentials method 176
- getFeatureBean method 162
- getFeatureOptionsBean method 163
- getFeatures method 163
- getfiledetails action 88, 159
- getfiledetails page. *See* file detail page
- GetFileDetailsActionForm class 159
- getFilter method 163
- getfolderitems action 89
- getfolderitems page. *See* folder index page
- getHomefolder method 163
- getImage method 161
- getIportalid method 44, 163

- getjobdetails action 86, 89, 160
- getjobdetails page. *See* request detail page
- getjobdetails.jsp 102
- GetJobDetailsActionForm class 160
- getLocale method 163
- getMaxJobPriority method 163
- getnoticejobdetails action 89
- getOnlylatest method 163
- getParameter method 43
- getPassword method 163, 173, 176
- getportletfolderitems action 89
- getProfile method 162, 164
- getProperty method 164
- getRepositoryType method 164, 176
- getrequesterjobdetails action 86, 89
- getRoleNames method 164
- getRunAsUser method 176
- getsavedsearch action 86, 92
- getServerUrl method 176
- getServerurl method 164
- getShowdocuments method 164
- getShowexecutables method 164
- getShowfolders method 164
- getSideBarFeatures method 164
- getSidebarSelected method 164
- getSkin method 161
- getSkinConfig method 164
- getSkinName method 164
- getStyle method 161
- getSubfeatures method 164
- getSystemname method 164
- getTimezone method 165
- getUserAgent method 165
- getUserHomeFolder method 177
- getUserid method 165
- getUserName method 173, 177
- getView method 165
- getVolume method 165, 177
- global style elements 49–59
- global variables 132
- Google translate elements 196
- goto action 86
- graphical user interfaces. *See* user interfaces
- graphics files. *See* image files
- graphics. *See* images
- GroupBean class 161
- GUIs. *See* user interfaces

H

- headline parameter
 - execute report page 104
 - output page 118
 - submit job page 128
- headlines. *See* headline parameter
- help 132
- help book list 199
- help collection directories 186
- help collections 185, 188, 191, 199
 - See also* help systems
- help content management files 187
- help content page title bars 198
- help content pages
 - accessing 185, 186
 - adding 201
 - changing additional links footer on 194–195
 - changing company logos on 193–196
 - changing content in 198, 199
 - changing Google translate element in 196
 - changing icons on 196
 - creating index entries for 204, 205
 - removing 200
- help directory 185, 186
- help files 184, 185, 189, 191
- help indexes 204, 205
- help keywords 204, 205
- help links 191, 192, 203
- help navigation frame 187
- help navigation pages 185, 196
- help system locations 184
- help systems 189, 198
- help topics 186, 188, 197, 199, 201
- help.js 132
- hexadecimal color values 57
- hexadecimal encoding 12
- home directory 23
- home folders 107, 147, 163, 166, 177
- home page 83, 93, 107
- homeFolder parameter 109
- homeFolder variable 148
- hosts 66
- HTML code 40, 46, 49
- HTML files 185
- html format value 138

- HTML tables 47, 48
- htmlselect.js 132
- HTTP requests 4, 8, 69, 162
- HTTPS requests 4
- hyperlinks
 - See also* URLs
 - changing targets for 18
 - creating 160
 - defining action paths and 43
 - displaying for specific locales 18
 - referencing files and 48
 - setting targets for 18
- hypertext markup language. *See* HTML code

I

- i18n tag library 143
- icon files 19, 59, 60, 74, 196
- icons
 - channels 117
 - features 19
 - help systems 196
 - replacing 52, 59
 - side menu 74
- IContentList interface 158
- ID parameter 102
- ID tag 74
- id variable 144, 149
- IDAPI_TIMEOUT parameter 68
- IDAPI. *See* Information Delivery API
- ifExists parameter 118, 128
- iHub
 - balancing workload on 6, 7, 31
 - connecting to 70, 79, 147, 167, 171
 - deploying Information Console and 4
 - getting security credentials for 176, 177
 - installing Information Console with 5, 68
 - logging in to 171
 - running Information Console and 14
 - sending requests over 8, 11, 24
- iHub releases 28
- iHub services 120, 122, 123
- iHub System 5, 22
- iHub system names 167
- image files 59, 161, 196
- Image tag 161
- imageid parameter 139
- images
 - adding background 59
 - changing 41, 52, 59–60
 - customizing 59
 - referencing 60
 - selecting skins and 49
 - uploading 161
- images directory 60
- immediate jobs 104, 128
- index pages 93, 108
- information 165
 - See also* data
- Information Console
 - accessing functionality 11, 42, 64, 72, 136, 158
 - adding web pages to 10, 43, 44, 48
 - adjusting layers for 132
 - building user interface for 4, 45, 49, 142
 - changing default settings for 30, 31, 32
 - changing deployed versions of 6
 - changing messages for 33, 34, 36
 - changing side menu for 74
 - changing web pages for 39, 48
 - configuring as web applications 29–39, 64
 - configuring proxy servers for 8, 9
 - creating context root for 27–29
 - creating custom applications for 8, 22, 27, 39–48
 - creating online help for 184–193, 198
 - customizing 5, 6, 49
 - deploying 4, 5, 6, 78
 - displaying information about 95, 110
 - displaying pages for 18, 82
 - grouping applications for 26
 - installing 5, 6, 68
 - localizing messages for 76–77
 - logging in to 12, 114, 172
 - logging out of 115
 - overview 4, 8
 - renaming default files for 48
 - resizing pages for 133
 - retrieving session information for 43
 - running multiple instances of 7, 27
 - selecting skin for 49, 51, 55
 - setting options for 109, 117
 - starting 8
 - viewing available locales for 75

- Information Console (*continued*)
 - viewing changes to 41
- Information Console custom pages. *See* web pages
- Information Console library 173, 179
- Information Console Security Extension 170
- Information Console technology 4
- Information Delivery API 4, 8
- init method 165
- input 34, 42
- INSTALL_MODE parameter 68
- installing Information Console 5, 6, 68
- instanceid parameter 139
- Interactive Viewer 17, 49, 80, 138
- Interactive Viewer help collections 186
- Interactive Viewer servlet 137
- InteractiveViewing subfeature 17, 75
- Intermediate functionality level 14, 15, 73
- internationalization. *See* locales
- internationalization tag library 142, 143
- internet applications. *See* web applications
- invokeSubmit parameter 104, 128
- IP addresses 171
- iportal context root 23
- iportal directory 24, 25
- iPortal Security Extension. *See* IPSE applications
- iPortalID parameter 84
- iPortalLogin action 89
- iPortalRepository class 44
- iPortalSecurityAdapter class 173, 174, 175
- IPSE applications 170, 173
- IPSE Java classes 173
- isAlwaysGetFolderList method 165
- isEnterprise method 44, 177
- isFileTypeAllowed method 178, 179, 181
- isHomeFolderSet method 165
- isIE method 162
- isNS4 method 162
- isNS6 method 162
- isnull parameter 104
- isShowFilters method 165
- isViewInNewBrowserWindow method 165
- IUploadSecurityAdapter interface 179, 181
- iv action 86, 89
- IVServlet. *See* Interactive Viewer servlet

J

- Jakarta Struts action mapping 82, 85
- Jakarta Struts code 46
- Jakarta Struts Framework 40, 43, 133
- Jakarta Struts templates 44, 45, 46
- Java classes 40, 173, 179
- Java Servlet API 40
 - See also* servlets
- JavaBean methods 158
- JavaBeans 43, 45, 57, 158, 165
- JavaBeans class reference 158
- JavaBeans package reference 158
- Javadoc 158
- JavaScript code 22, 40
- JavaScript components 132
- JavaScript files
 - accessing 44, 46
 - changing 42
 - creating online help and 185, 187
 - developing web applications and 132
 - referencing images and 60
- JavaScript reference 132
- JavaServer Pages. *See* JSPs
- job action forms 160
- job classes 160
- JobActionForm class 160
- jobID parameter
 - delete job page 99
 - delete status page 99
 - request detail page 102
 - request drop page 103
 - requests index page 110
- jobName parameter
 - delete job page 99
 - delete status page 100
 - execute report page 104
 - request drop page 103
 - schedule page 125
 - submit job page 128
- JobPriority subfeature 17, 75
- jobs
 - See also* requests
 - canceling 103
 - deleting 99, 103
 - displaying 97, 110, 123

- executing 10
- filtering 110
- getting information about 101, 160
- listing pending 120, 125
- removing notifications for 99
- running immediately 104, 128
- scheduling 84, 96, 124, 128
- sending notifications for 17, 110, 128
- setting priorities for 17, 105, 129, 166
- setting properties for 161
- submitting 10, 74, 108, 126, 161
- viewing parameters for 120
- Jobs attribute (features) 16, 74
- Jobs link 19
- jobs pages 110
- jobState parameter 99, 100, 103
- JSP code 46, 142
- JSP custom tag reference 142, 143, 144
- JSP engine 5, 8, 26
- JSP extensions 40
- JSP file names 48
- JSPs
 - accessing 24
 - changing templates and 46
 - compiling 22
 - customizing 24, 44, 45, 46
 - displaying 42
 - generating web pages and 10–11, 22, 48
 - getting input from 42
 - getting session information and 43
 - implementing URIs and 23
 - implementing URLs and 22
 - linking styles in 56, 57
 - locating specific 42
 - mapping actions to 82, 87
 - naming 82
 - referencing images in 60
 - selecting templates for 24, 45
 - updating changes to 41
- JUL_LOG_CONSOLE_LEVEL parameter 68
- JUL_LOG_FILE_COUNT parameter 68
- JUL_LOG_FILE_LEVEL parameter 68
- JUL_LOG_FILE_SIZE_KB parameter 68

K

- key attribute 149

L

- label keys 18, 74
- Labelkey tag 74
- labels 34, 74, 76
- landing page 24, 40–41, 44
- language codes 76, 77
- languages 64
- language-specific applications. *See* locales
- large icons 19, 74
- LargeIcon tag 19
- LaunchHelp method 190
- launchiv parameter 139
- layer functionality 132
- layer.js 132
- Level tag 18, 73
- libraries 44, 142, 173
- license page 110
- limit parameter 105
- limitNumber parameter 105
- Link tag 18, 57, 74, 160
- LinkBean class 160
- linking style definitions 56, 57
- linking to files 48
- linking to folders 107
- linking to web pages 47, 48
- links 8, 41, 72, 107, 188
 - See also* hyperlinks
- Linux systems 5, 23, 36, 38
- list package 158
- list pages 83, 94, 111
- lists 67, 68, 111, 113, 158
- load balancing 6, 7, 31
- locale codes 76, 77
- locale IDs 76
- locale names 76
- locale parameter 84, 139
- Locale tag 76
- locales
 - accessing repository for 22
 - accessing resources for 18, 144, 149
 - adding 75
 - changing 118
 - configuring 75, 76
 - creating error messages for 76–77
 - formatting data for 146
 - providing help topics for 188, 191

- locales (*continued*)
 - setting default 30, 31, 66, 76, 165
 - setting global styles for 49
 - specifying current 84, 144
 - specifying preferred 144
 - translating reporting applications for 33
- localhost value 11
- localizing messages 33, 76–77
- log file numbers 68
- log files 67, 68
- LOG_FILE_LOCATION parameter 68
- logging in to
 - iHub 171
 - Information Console 12, 114, 172
- logging levels 68
- login action 9, 86, 89
- login banner page 83, 94, 114
- login credentials 147, 175, 176
- login forms 162
- login information 66, 162, 172
 - See also* login credentials
- login page 12, 44, 83, 94, 114
- login requests 171
- login tag 143, 147
- login tag library 143
- LOGIN_TIMEOUT parameter 68
- LoginForm class 162
- loginPostBack parameter 115
- logins
 - customizing 170
 - forcing 84
 - getting user information for 162, 176, 177
 - redirecting 9, 115
- LoginTag class 147
- logos 59, 193–196
- logout action 86, 90
- logout page 83, 94, 115

M

- machine names 11
- magnifying glass icon 60
- Management Console 5, 16, 136
- MAX_BACKUP_ERROR_LOGS
 - parameter 68
- MAX_CONNECTIONS_PER_SERVER
 - parameter 79

- MAX_LIST_SIZE parameter 68
- MDS_ENABLED parameter 7, 31, 79
- MDS_REFRESH_FREQUENCY_SECONDS
 - parameter 31, 79
- MDS. *See* Message Distribution service
- memory 67, 79
- menus 16, 60, 74, 109, 132
- Message Distribution Service 6, 31, 79, 121
- message keys 34
- message tag 143, 149
- messages 33, 34, 36, 108
 - See also* error messages
- MessageTag class 149
- method calls 165
- methods 132, 158
- Microsoft Windows. *See* Windows systems
- Mobile attribute (features) 16, 74
- mobile viewing 74
- MOBILE_APP_DOWNLOAD parameter 66
- mode parameter 122
- Model-View-Controller architecture 22
- multi-byte characters 14
- multilingual reporting. *See* locales
- My dashboard page 115
- My Documents folder 109
- My Documents link 109
- My Documents page 49
- My Folder icon 59
- My Folder link 107

N

- name parameter
 - DownloadFile servlet 137
 - file or folder detail page 101
 - file or folder drop page 102
 - privileges page 123
- names (action paths) 48
 - See also* file names; user names
- naming
 - functionality levels 16, 73
 - JSPs 82
 - output files 105, 119, 128
 - skins 53
- naming restrictions 22, 82, 127, 136, 142
- NAT routers 171
- native2ascii utility 77

- Network Address Translation (NAT) 171
- networks 4, 6, 69, 170, 171
- new request index page 93, 108
- newKey parameter 118
- newLocale parameter 118
- newTimeZone parameter 118
- notification page 83, 94, 116
- notifications
 - deleting 99
 - generating 117
 - sending 17, 110, 128
 - setting options for 116
 - specifying user names for 105, 110
 - storing 160
- notificationSupported parameter 128
- notify parameter 128
- null values 104

O

- object aging. *See* archiving
- objectID parameter 101
- objects 142
- oldKey parameter 118
- onceDate parameter 125, 128
- onceTime parameter 125, 128
- on-demand paging. *See* immediate jobs; progressive viewing
- online documentation vii
- online help 184–193, 198
 - See also* help; online documentation
- onlyLatest parameter 114
- open source frameworks 40
- opening
 - help files 185
 - Interactive Viewer 17
 - login page 84
 - skin manager 51
 - web applications 27, 108
 - web browser windows 118, 168
- operating systems. *See* UNIX systems; Windows systems
- option controls 132
- options action 90
- Options functionality 109
- options index page 94, 109
- options JSPs 11

- options page 83, 94, 117, 162
- options save action 90
- output 105, 118, 128
- output file names 120
- output files
 - See also* report files
 - deleting 104, 127
 - limiting number of 105
 - naming 105, 119, 128
 - saving 105, 118
- output page 83, 94, 118
- outputFolderType parameter 105, 119
- outputFormat parameter 128
- __outputName parameter 128
- outputName parameter
 - execute report page 105
 - output page 119
 - submit job page 128
- overwrite parameter 105

P

- packages 158
- page breaks 66
- page engine 8
- page icons 197
- page names 82
- page not found messages 108, 119
- page not found page 83, 94, 119
- page parameter 139
- page ranges 139
- page styles 139
- parameter components 57
- parameter definitions 30
- parameters
 - adding to URIs 12, 13, 84
 - changing 30
 - configuring Information Console and 64
 - connecting to Encyclopedia and 22, 79
 - customizing reporting applications and 30
 - displaying 120, 128
 - loading JSPs and 22
 - loading web pages and 10, 12
 - referencing report 127
 - returning session information and 43
 - running reports and 80, 104, 106
 - setting up report viewers and 80

- parameters list 120
- parameters page 57, 83, 94, 120
- partitionName parameter 122
- partitions 122
- password parameter 85
- passwords
 - adding to URIs 85
 - getting 163, 176
 - setting 166
 - updating 117, 172
- paths
 - context roots 43
 - dashboard resources 79
 - home folders 107
 - icons 19
 - image files 60
 - log files 68
 - temporary files 69, 80
- payloadSize parameter 122
- pdf format value 138
- pending jobs 120, 125
- pending page 83, 94, 120
- performance 67
- performance analytics gadgets 65
- ping action 90, 160
- ping modes 122
- ping page 84, 94, 120
- PingActionForm class 160
- PMD. *See* Process Management Daemon
- pool. *See* connection pools
- pop-up menus 132
- popupmenu.js 132
- ports 8, 11
- postback parameter 128
- ppt format value 138
- preferences 84
- prefix attribute 142
- PRELOAD_ENGINE_LIST parameter 68
- presentation models 22
- previewing application pages 41
- Print icon 197
- print page 84
- prioritizing jobs 17, 105, 166
 - See also* priority parameter
- priority parameter
 - execute report page 105
 - submit job page 129

- priority settings. *See* priorityValue parameter
- priorityValue parameter
 - execute report page 105
 - submit job page 129
- private cache 65
- private networks 171
- privileges 5, 123, 127, 147, 159
- privileges page 84, 94, 123
- Process Management Daemon 115
- process redirect page 9
- processed action status page 103
- processID parameter 122
- ProfileBean class 162
- ProfileName parameter 70
- profiles 32, 70, 80, 162, 166
- programmers 4
- progressive parameter 105, 129
- progressive viewing 69, 105, 129
- PROGRESSIVE_REFRESH parameter 69
- PROGRESSIVE_VIEWING_ENABLED
 - parameter 69
- prompts 34
- properties 166
- properties files 34
- protecting data 170
 - See also* security
- proxy servers 7, 8, 9, 69, 171
- PROXY_BASEURL parameter 69
- ps format value 138
- public skins 51

R

- recurringDay parameter
 - execute report page 105
 - schedule page 125
 - submit job page 129
- recurringTime parameter 125, 129
- redirect attribute 9
- redirect parameter
 - delete job page 99
 - delete status page 100
 - file or folder drop page 102
 - options page 118
 - request drop page 103
 - submit job page 129
- redirection 9, 66, 115, 128, 189

- referencing
 - files 48
 - images 60
 - report parameters 127
- refresh intervals 32, 79
- related topics icon 197
- relative hyperlinks 48
- removing. *See* deleting
- renaming files 48, 60
- report design area 58
- report document files 100, 114, 167
- report emitters 65
- Report Encyclopedia. *See* Encyclopedia volumes
- report executable files 114, 120, 167
 - See also* executable files
- report execution requests. *See* jobs; requests
- report files
 - See also* specific type
 - accessing 10, 74, 112, 159
 - archiving 104, 127
 - assigning privileges to 123
 - copying 146
 - deleting 17, 98, 102
 - displaying 114
 - downloading 137
 - filtering 112
 - getting information about 100, 159
 - linking to 48
 - overwriting 105, 118, 128
 - saving 105, 118, 133
 - searching 125
 - sharing 17
- report libraries 142, 173
- __report parameter 139
- report parameters 120, 127, 128
- report server. *See* iHub
- report viewers 80, 82, 130, 137
 - See also* specific viewer
- report.js 132
- reporting applications. *See* applications
- reporting services. *See* iHub services
- reporting system. *See* iHub System
- reportlet format value 138
- reports
 - deploying 5
 - displaying 10, 11, 24, 69, 80, 132, 137
 - downloading 137
 - filtering 112
 - refreshing 69
 - running 67, 103, 125
 - saving 105, 118
 - searching 74
 - submitting requests for 8, 103
- repositories
 - See also* Encyclopedia volumes
 - accessing items in 10, 74, 112
 - displaying information about 47
 - downloading from 139
 - getting type 176
 - returning type 44
- REPOSITORY_CACHE_TIMEOUT_SEC parameter 79
- repositoryType parameter 139
- request detail page 93, 101
- request drop page 93, 103
- Request page 118
- request variable 43
- requestercanceljob action 86, 90
- requesterdeletejob action 86, 90
- requesterdeletejobschedule action 86, 90
- requestFilters parameter 118
- requests
 - See also* jobs
 - dropping 103
 - failing 117
 - limiting number of items returned 68
 - loading web pages and 10, 12
 - running multiple applications and 7
 - sending 6, 11, 22, 24, 27
 - specifying action paths for 48
 - submitting 8, 96, 103, 125, 171
 - testing Encyclopedia and 122
 - timing out 66
- requests index page 94, 109
- resetFilter parameter 110, 114
- resize.js 133
- resource bundles 144, 149
- resource files 18, 33, 76
- resources 10, 18, 22, 46, 70, 132
- restarting Apache Tomcat service 29
- reverse proxies 9
- RGB color values 52, 57
- rgb method 57

- Role tag 16, 73
- roles
 - creating 14, 167
 - defining functionality levels and 14, 16
 - setting privileges for 127
- rtl parameter 139
- run requests. *See* jobs; requests
- running
 - Java servlets 136
 - jobs 10
 - reports 67, 103, 125
- running page 84, 94, 123

S

- saveas.js 133
- saveOutput parameter 105
- savesearch action 92
- saving
 - dashboards 79
 - image files 161
 - output files 105, 118
 - report files 133
- schedule page 84, 94, 124
- schedule properties 124
- scheduled job page 84, 94, 125
- schedulePeriod parameter 129
- scheduleType parameter 125, 129
- scheduling jobs 84, 96, 124, 128
- Search attribute (features) 16, 74
- search expressions 126
- search file page. *See* file list page
- search folders page 84, 95, 125
- Search link 15, 19
- Search page 159
- search results 49
- searchfiles action 90, 91, 159
- searchfiles page. *See* file list page
- SearchFilesActionForm class 159
- searchFilter parameter 126
- searching
 - folders 74, 125
 - reports 74
- security 4, 65, 170, 179
- security adapter class 171, 173, 174, 175
- security adapters 171, 172–182
- security extension Java classes 179

- security manager 175
- security roles. *See* roles
- SECURITY_ADAPTER_CLASS
 - parameter 69, 175
- selectchannels action 90
- selectchannels page. *See* channels list page
- selectjobnotices action 90, 160
- selectjobnotices page. *See* channel contents list page
- SelectJobNoticesActionForm class 160
- selectjobs action 90, 161
- selectjobs page. *See* requests index page
- SelectJobsActionForm class 161
- SelfNotificationWithAttachment
 - subfeature 17, 75
- sending notifications 17, 110, 128
- sending requests 11, 22, 24, 27
- server clusters. *See* clusters
- server parameter 123
- server profiles 32
- server URLs 167
- servers
 - See also* iHub
 - accessing 44, 170
 - balancing workload among 7, 31
 - configuring context root for 27
 - deploying Information Console over 4, 5, 6, 170
 - dropping connections to 66
 - extending functionality of 136
 - installing Information Console on 5
 - maintaining session states for 7
 - optimizing performance for 67
 - restarting 28, 29
 - retrieving session information for 43
 - running multiple applications and 7, 27
 - sending requests over 8, 10
 - setting default 32
 - setting up firewalls and 8, 9, 170
 - updating user settings for 117
- serverURL parameter
 - execute report page 105
 - Interactive Viewer servlet 139
 - submit job page 129
 - URIs 85
- services 120, 122, 123
- servlet engine 5, 8, 27, 41

- servlet names 136
- servlets 10, 22, 136
- servlets reference 137
- session attributes 113
- session information 43, 115
- session state 7
- session variables 113
- SESSION_DEFAULT_PARAMETER_VALUE_ID parameter 69
- sessions 7, 66, 68, 69
- sessionTimeout parameter 69
- setAcLocale method 165
- setAlwaysGetFolderList method 165
- setAuthid method 166
- setCurrentfolder method 166
- setDefaultServerURL method 166
- setDefaultVolume method 166
- setFeatureOptions method 166
- setFilter method 166
- setHomefolder method 166
- setMaxJobPriority method 166
- setOnlylatest method 166
- setPassword method 166
- setProfile method 166
- setProperty method 166
- setRequest method 162
- setRoleNames method 167
- setServerurl method 167
- setShowdocuments method 167
- setShowexecutables method 167
- setShowFilters method 167
- setShowfolders method 167
- setSideBarFeatures method 167
- setSidebarSelected method 167
- setSkinConfig method 167
- setSkinName method 167
- setSystemname method 167
- setTimezone method 168
- setUserAgent method 168
- setUserid method 168
- setView method 168
- setViewInNewBrowserWindow method 168
- setVolume method 168
- ShareFile subfeature 17, 75
- showDocument parameter 114
- showExecutables parameter 114
- showFolders parameter 114
- side bars. *See* side menu
- side menu 16, 60, 74, 109
- simple object access protocol. *See* SOAP messages
- skin classes 161
- skin descriptions 54
- skin editor 161
- skin manager 39, 40, 51, 54, 161
- skin names 167
- SkinBean class 161
- skincustomization.js 133
- skinedit action 91
- SkinEditorActionForm class 161
- skinerror action 86
- SkinManagerActionForm class 161
- SkinManagerInfoBean class 161
- skins
 - accessing templates for 45
 - adding background images to 59
 - applying style definitions to 56, 57
 - changing application pages and 40, 49
 - changing images and 59
 - changing side menu for 74
 - cloning 51, 52, 53
 - creating 51, 52, 53, 167
 - customizing 49, 51, 52, 53, 74, 161
 - deleting 51
 - editing 161
 - maintaining 133
 - naming 53
 - previewing 51, 52
 - selecting 49, 51, 55
 - specifying default 51, 55
- small icons 19, 74
- SmallIcon tag 19
- SOAP messages 36
 - See also* requests
- source code 44
- space character 13
- special characters 12
- Standard Viewer 137
- start folders 147
- startFolder variable 148
- starting Information Console 8
- startup messages 108
- startUpMessage parameter 108
- string substitution 18

- string tag 143
- stringList tag 143
- strings 37, 84, 106
- Struts action mapping 82, 85
 - See also* Jakarta Struts Framework
- strutscommon.js 133
- style definition files 56
- style definitions 56, 57
- style sheets
 - accessing 44, 46
 - changing styles in 57
 - customizing web pages and 40, 59
 - linking to JSPs 56, 57
 - specifying color settings in 57
 - updating changes to 42
 - viewing changes to 58
- STYLE tag 56, 57, 161
- styles 56–59
 - See also* style sheets
- subdirectories 24
- SubfeatureID tag 17, 75
- subfeatures 17, 75
- subfolders 126
- submit job page 84, 95, 126
- submitJob action 91, 161
- submitJob page. *See* submit job page
- SubmitJobActionForm class 160, 161
- submitting jobs 10, 108, 126, 161
- subpage parameter 108, 110
- subscribeChannel action 91
- SubscribeChannel subfeature 17, 75
- SubscribeChannelActionForm class 159
- succComp parameter 118
- succEmail parameter 118
- synchronizing data 17
- system. *See* iHub System
- system names 167
- system resources. *See* resources

T

- Tab class 150
- tab tag 143, 150
- tabbed dialogs
 - See also* tabs
 - defining tabs for 150, 153, 155
 - setting orientation of 154
 - specifying content for 145, 153
- tabbed property sheets 109
 - See also* tabbed dialogs
- Tabbed skins 50
- TabBegin class 151
- tabBegin tag 143, 151
- TabEnd class 151
- tabEnd tag 143, 151
- table of contents (help topics) 199, 201, 203
- TABLE tag 47, 48
- tableList action 91
- tables. *See* HTML tables
- TabMiddle class 152
- tabMiddle tag 143, 152
- TabMiddleSelected class 153
- tabMiddleSelected tag 143, 153
- tabPanel tag 143, 153
- TabPanelTag class 153
- tabs
 - See also* tabbed dialogs
 - associating pages with 153
 - choosing skins and 50
 - defining adjacent pairs 155
 - defining labels and keys for 150
 - moving focus to 154
 - selecting 109
 - setting attributes of 153
 - setting order of 151, 152
 - specifying as default 154
- TabSeparator class 155
- tabSeparator tag 143, 155
- tag libraries 44, 142
- tag library descriptor 142
- tag lines. *See* headline parameter
- tag names 142
- taglib directive 142
- tags
 - adding locales and 75
 - adding time zones and 76
 - changing help topics and 198
 - defining features and 18, 74
 - defining functionality levels and 16, 73
 - defining subfeatures and 17, 75
 - encapsulating frequent tasks and 142, 143
- targetPage parameter 12, 115
- temp directory 80
- TEMP_FOLDER_LOCATION parameter 80

- template element 45
- template files 45
- template tags 45
- templates
 - accessing 45
 - building JSPs and 24, 45, 46
 - changing landing page and 41
 - creating web pages and 48
 - customizing applications and 44–48
 - selecting 45
- temporary files 69, 80, 122
- temporary server profiles 72
- testing
 - applications 27
 - help links 191, 192
 - reporting services 123
- text 33, 34, 37, 45, 76, 198, 204
- text files 77
- text messages 34, 36, 108
- third-party applications 7
- time stamps 106, 122
- time zone IDs 76
- time zones 30, 31, 64, 66, 76, 85, 118, 168
- time-out values 66, 68, 69
- timeToDelete parameter
 - execute report page 105
 - submit job page 129
- timezone parameter 85
- TimeZone tag 76
- title bars 59
- TITLE_LANDING_PAGE parameter 34
- titles 34, 198
- TLD files 142
- toolbars 59, 133
- toString method 160, 165
- transient files 69, 80, 122
- TRANSIENT_STORE_MAX_SIZE_KB
 - parameter 69
- TRANSIENT_STORE_PATH parameter 69
- TRANSIENT_STORE_TIMEOUT_SEC
 - parameter 69
- translating reporting applications 33
- treebrowser action 91
- Treeview skins 50
- truncated strings 84
- trusted names 11

U

- unauthorized users 170
- unencode method 132
- Uniform Resource Identifiers. *See* URIs
- Uniform Resource Locators. *See* URLs
- UNIX systems 5, 23, 36, 38
- updatechannel action 91
- updating
 - data 17
 - passwords 117, 172
 - user options 117
 - web pages 41
- upgrades 20, 52
- upload security adapter class 179, 180
- upload security adapter interface 179, 181
- upload security adapters 178–181
- UPLOAD_FILE_TYPE_LIST parameter 69, 178
- UPLOAD_SECURITY_ADAPTER
 - parameter 180
- UPLOAD_SECURITY_MANAGER
 - parameter 69
- uploadimage action 91
- uploading binary files 136
- uploading image files 161
- uploadlicense action 91
- uri attribute 142
- URIs
 - accessing reporting applications and 27
 - adding parameters to 12, 13, 84
 - creating 11, 12
 - displaying feature-specific pages and 18
 - encoding characters and 12, 14
 - implementing 23
 - loading servlets and 136
 - locating specific JSPs and 42
 - obtaining list values and 113
 - overview 11, 82
 - Process Management Daemon and 115
 - redirecting logins and 115
 - redirecting web pages and 9
 - referencing in tag libraries 142
 - returning diagnostic information and 121
 - running reports and 104, 106
 - submitting requests and 8, 11, 24
 - viewing reports and 130

URIs reference 92, 130

URLs

- activating security manager and 172
- connecting to iHub System and 22, 70, 167
- opening help files and 188, 189, 190
- redirecting web pages and 9, 66
- setting up firewalls and 8
- specifying default 166
- transmitting actions and 22, 43

user authentication. *See* authentication

user classes 162

user IDs 85, 139, 168

user interfaces

- building 4, 45, 49, 142
- changing elements in 59–60
- enabling features for 14, 16, 72
- enabling subfeatures for 17, 75
- submitting requests and 8

user names 177

See also userName parameter

user option settings 11

user parameter 115

user profiles 162, 166

user-agent header 162

UserAgentBean class 162

UserAgentBean objects 168

userID parameter 85

userid parameter 139

UserInfoBean class 43, 162, 162–168

userName parameter

- delete status page 100
- options page 118
- request detail page 102
- requests index page 110

UserOptionsActionForm class 162

users

- accessing home page 107
- displaying current settings for 107
- displaying preferences for 84
- getting authentication IDs for 44
- getting home folders for 177
- getting passwords for 163, 176
- getting security credentials for 176
- returning information about 162
- selecting functionality levels 162
- sending notifications to 105, 110, 116, 128
- setting default skins for 51, 55
- setting features for 74, 167
- setting functionality levels for 14, 15, 16, 72, 73
- setting passwords for 166
- updating passwords for 117, 172
- updating settings for 117
- validating credentials for 175
- viewing subscribed channels for 111, 117, 158

users parameter 105

users tag library 142

UTF-8 encoding 14

V

values. *See* data

variables 34, 37, 113

verifyFile method 178, 179, 181

version names. *See* versionName parameter

version parameter

- DownloadFile servlet 137

- file or folder detail page 101

versionName parameter

- execute report page 106

- output page 119

- submit job page 129

view constants 168

View service 123

viewer getsavedsearch action 92

viewer page 84, 95

viewer savesearch action 92

viewer servlet 137

viewers 80, 82, 130, 137

See also specific Actuate viewer

viewframeset action 86

viewing

- banners 95, 114

- color settings 55

- completed jobs 97

- current jobs 123

- data 22, 48

- error messages 103, 119

- failed jobs 110

- files and folders list 67

- folders 114, 167

- help topics 201, 203

- locales 75

- login page 12, 114
- pending jobs 125
- report executables 114, 167
- report parameters 120, 128
- reports 10, 11, 24, 69, 80, 132, 137
- repository information 47
- search results 49
- subscribed channels 96, 111, 117, 158
- web pages 42
- viewnav.js 133
- viewNewBrowser parameter 118
- viewpage action 86
- views 168
- volume icons 59
- volume parameter
 - execute report page 106
 - Interactive Viewer servlet 139
 - submit job page 129
 - URIs 85
- volume profile name 32
- volume profiles 70–72, 80, 177
- VOLUME_PROFILE_LOCATION
 - parameter 80
- VolumeProfile parameter 85
- VolumeProfiles tag 70
- volumes. *See* Encyclopedia volumes

W

- wait parameter 106
- wait values 66, 67, 106
- waitforreportexecution action 92
- WAR deployments 78
- WAR files 5, 6
- web applications 8, 40, 170
 - See also* applications
- web archive files. *See* WAR files
- web browsers
 - changing style definitions for 57
 - changing title bar text for 198
 - changing web pages and 48
 - deploying Information Console and 5
 - detecting 132, 162
 - displaying environment settings for 107
 - encoding and 12, 14
 - issuing URIs and 84
 - loading web pages for 10, 12
 - maintaining session state for 7

- opening new windows for 118, 168
- preserving login information for 66
- redirecting 9, 66, 115, 128
- setting cache for 65
- specifying 168
- web pages
 - adding 10, 43, 44, 48
 - associating with tag libraries 142
 - caching 41, 65, 79
 - creating banners for 95
 - customizing 39, 46, 48
 - displaying 42
 - formatting 48
 - generating 10–11, 22, 48
 - linking to 47, 48
 - loading 10, 12
 - localizing 144, 149
 - navigating through 47
 - resizing 133
 - submitting requests and 8
 - updating 41
 - viewing changes to 41
- web resources 10, 22, 46
- web servers 5
 - See also* servers
- web sites 5
- web.xml 64
- WEB-INF directory 64, 142
- welcome text 41
- wildcards 126
- window displays 132
- Windows language pack 77
- Windows systems 5, 23, 35, 37
- working folders 96, 113
- workingFolder parameter 96
- workingFolderID parameter 98
- workingFolderName parameter 98
- wr action 86
- write tag 57
- wwhelp directory 185

X

- xls format value 138
- XML files 41
- XML tag reference 142, 144
- XML tags 142

